# A comparison and evaluation of approaches to the automatic formal analysis of musical audio

Jordan B. L. Smith

Master of Arts

Music Technology Area
Department of Music Research
Schulich School of Music

McGill University
Montreal, Quebec, Canada

31 August 2010

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Arts

# Acknowledgements

Special thanks are owed to several esteemed colleagues at other institutions who provided their algorithms or who participated in the evaluation. Luke Barrington at the University of California, San Diego provided the code for his algorithm (Barrington et al. 2009), with assistance from Emanuele Coviello. The evaluation of Xavier Janer Mestres' (2007) segmentation algorithm was carried out by Nicolas Wack and Emilia Gómez at Universitat Pompeu Fabra. Ewald Peiszer has made his algorithm (2007) and many of his previous results available online. Mark Levy has implemented his algorithm (Levy and Sandler 2008) as a Vamp plugin for Sonic Visualiser, both of which are free to download. Finally, Tristan Jehan and Brian Whitman, founders of Echonest, allow free public use of their analysis tools via a developer API. I am grateful for all of the above researchers and inspired by their willingness to share.

I am likewise indebted to those institutions who have freely shared structural annotations for large corpora of music, including the Real World Computing project; the Centre for Digital Music at Queen Mary, University of London; Universitat Pompeu Fabra; and Tampere University of Technology.

This work was supported by a Joseph-Armand Bombardier Canada Graduate Master's Scholarship from the Social Sciences and Humanities Research Council of Canada, and by a Master's scholarship from the Fonds du recherche sur la société et la culture.

This thesis is dedicated to my advisor Ichiro Fujinaga, who introduced me to Music Information Retrieval and who has been an invaluable teacher and guide to me for the past two years. I must also extend my sincerest thanks to my colleagues, advisors, friends and family who have sustained me throughout the creation of this work.

# Abstract

Analyzing the form or structure of pieces of music is a fundamental task for music theorists. Several algorithms have been developed to automatically produce formal analyses of music. However, comparing these algorithms to one another and judging their relative merits has been very difficult, principally because the algorithms are usually evaluated on separate data sets, consisting of different songs or representing wholly different genres of music, and methods of evaluating the performance of these algorithms have varied significantly. As a result, there has been little benchmarking of performance in this area of research. This work aims to address this by directly comparing several music structure analysis algorithms.

Five structure analysis algorithms representing a variety of approaches have been executed on three corpora of music, one of which was newly assembled from freely distributable music. The performance of each algorithm on each corpus has been measured using each of an extensive list of performance metrics.

# Abrégé

Faire une analyse de la structure d'un pièce de musique est une tâche de première importance pour les théoriciens de musique. Ainsi, plusieurs algorithmes qui visent à produire de telles analyses automatiquement ont été développés. Il est toutefois difficile de comparer ces algorithmes parce qu'ils sont souvent évalués sur des corpus provenant de différentes œuvres musicales ou même de différents genres de musique, et la façon d'en faire l'évaluation a varié beaucoup. En conséquence, peu d'analyses comparatives de la performance de ces algorithmes ont été effectuées. Ce travail vise à aborder ce problème en comparant directement plusieurs algorithmes de l'analyse de structure musicale.

Cinq algorithmes représentant une gamme d'approches à l'analyse de structure musicale ont été exécutés sur trois corpus de musique, dont un créé de sources publiques afin qu'il soit libre à partager. La performance de chaque algorithme sur chaque corpus a été évalué à l'aide de plusieurs mesures standard.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Motivation: Issues in evaluating structure analysis algorithms

In recent years, there has been a surge in interest in developing algorithms to automatically analyze the structure of pieces of music. This recent research push began roughly ten years ago with the publication of a novel method to visualize structure using self-similarity matrices (Foote 1999). Within years, dozens of researchers had proposed extensions to this technique, some borrowing sophisticated filtering and thresholding techniques from the field of image analysis (e.g., Lu et al. 2004; Eronen 2007). Others have taken up the problem of structure analysis in new ways, employing statistical approaches involving Hidden Markov Models, or other intelligent grouping methods (e.g., Logan and Chu 2000; Abdallah et al. 2006; Levy and Sandler 2008). Still others have proposed algorithms to address special musical situations, such as transpositions and tempo changes (e.g., Goto 2003a; Müller and Clausen 2007) or hierarchical relationships (e.g., Jehan 2005b; Rhodes and Casey 2007).

With such a diversity of approaches, it is desirable to have some means of easily comparing them, to judge which approach is able to produce the most satisfactory results for a particular application. Unfortunately, two important factors have made such comparisons very difficult to make.

Firstly, few algorithms have been evaluated on the same corpora of music. This is partly because some algorithms are designed to address vastly different genres of music (for instance, classical vs. popular music), but a more important reason is that it is usually illegal for researchers to share their music collections with one another. As a result, it is common for researchers to produce their own corpora.

Secondly, studies have frequently reported different evaluation measures. This is partly because there has been little agreement about which evaluation measures are the most indicative of an algorithm's ability to describe musical structure, and partly because different studies have had different applications in mind, motivating different evaluation methods.

These factors are both related to a third problem: that producing ground truth annotations, against which the results of each algorithm are to be compared, is a highly

subjective and time-consuming process. If two different researchers are not able to share their music collections and must each collect their own, they may also need to produce their own annotations for their music. If their annotations differ in a significant way (for instance, if the vocabulary used in one set of annotations is highly restricted, whereas another is more open-ended), then the reported results cannot be easily compared, even if the music is very similar or identical. The result is that despite the large number of studies about automatic structure analysis, many of these studies rely on very small test sets, and the results of these studies are difficult to compare.

Despite these obstacles, some comparative evaluations have been published. These are usually studies that propose one or several new algorithms and compare their performance to each other (e.g., Levy and Sandler 2008) or to a single previous one (e.g., Barrington et al. 2009). However, such studies are unable to remove these obstacles for future studies: although they provide a means to compare two existing algorithms, it remains just as difficult as before for a researcher to determine how their new algorithm compares against previous ones, or how it would fare according to a new evaluation metric.

This is not an unusual situation in the field of Music Information Retrieval (MIR), where non-uniform evaluation, a lack of readily available ground truth, and restrictions on sharing music pose obstacles for researchers. The Music Information Retrieval Evaluation Exchange (MIREX) was founded in part to address these shared issues. This community-led event annually establishes benchmarks for MIR tasks by collecting algorithms from participants, running them on standardized test sets, and using standardized evaluation measures to analyze the results (Downie 2008). Although this solves the copyright issue of distributing music, the need to keep the data sets private still prevents researchers from experimenting with the data themselves. By contrast, data sets for the Text Retrieval Conference (TREC), on which MIREX was modeled, are freely available online at their website.[1]

In 2009, MIREX conducted its first comparative evaluation of structural segmentation algorithms. This was a landmark evaluation for this MIR task, but did not

[1] <http://trec.nist.gov/data.html> accessed 12 July 2010.

resolve all of the issues described above: a minor issue is that the size of the evaluation (the number of pieces totalled 297), though large in comparison to previous structure analysis studies, remains small in comparison to evaluations for other MIR tasks. But more importantly, the corpus of music used remains unsharable, as described above, as well as undisclosed. It can be guessed from the MIREX webpage devoted to the structural segmentation task, collaboratively written by the community submitting the algorithms, that the test set at least included the full studio catalogue of the Beatles (180 songs), but the identity of the songs in the test set is not public information. (In addition, two of the five algorithms tested at MIREX were anonymous submissions, so it is also unknown what approaches were being compared.)

As a result, we can only learn how the submitted algorithms rated against each other globally; we cannot learn about how successfully each algorithm handled specific musical situations. In fact, the same is true of virtually all structure analysis studies, for although global performance measures are always reported, only a few examples of any algorithm's output are shown. As an exception to the rule, all of the structural descriptions generated by the algorithms proposed in Ewald Peiszer's (2007) thesis are published online. Since different algorithms approach the task of structure analysis in different ways, it could be very interesting to examine their performance in specific musical situations, but current reporting practices to not facilitate this.

To maintain an open, repeatable, and easily updatable benchmark system for music structure analysis, researchers would ideally test algorithms on publicly available and freely distributable music, with the annotations and with the full output of each algorithm saved for future reference. This way, a future need for a new evaluation measure could easily be accommodated, and benchmarks would be easily updated in the case that any annotations were altered.

## 1.2 Description of the present work

This thesis is a comparative study of automatic structural analysis algorithms that aims to redress some of the issues associated with previous evaluations mentioned in Section 1.1. Firstly, this evaluation is carried out on new corpora drawn from public-domain sources that are freely distributable online. Secondly, the full results of the

evaluation are published (reporting the output of an extensive list of previously used evaluation metrics); each algorithm's full output is published as well, enabling future studies to be fairly compared with this one.

Five previously developed algorithms have been collected, representing a variety of approaches to the task of automatic structural analysis, and evaluated on three musical corpora. The three corpora have been selected to facilitate future reference to this benchmark evaluation: two corpora (Beatles and Real World Computing, or RWC) consist of sets of music and accompanying annotations that, although not freely distributable, have been subject to the most widespread testing so far in structure analysis research. This helps establish a link between the present evaluation and previously published evaluations. The other corpus, containing classical and jazz music, has been assembled from collections of public-domain or uncopyrighted audio which is freely available online. This way, other researchers may legally obtain this corpus at no cost, maximizing the potential for future comparative analysis. Structural annotations for these works have been generated for this thesis and are released into the public domain as well. Together, these corpora represent a diverse set of genres, including popular, classical, and jazz music.

The performance of each algorithm on each corpus has been measured using each of an extensive list of performance metrics. In the absence of a consensus regarding which evaluation measures best express the performance ability of analysis algorithms, all measures that have been used in previous publications on this topic have been reported for the present evaluation. In addition, the structural descriptions estimated by each algorithm have been published as an online appendix to this thesis,[2] enabling future researchers without access to the algorithms tested here to fully update this benchmark evaluation with new evaluation metrics.

## 1.3 Scope of the present work

Form and structure are fundamental concepts in music theory and analysis, so it is no surprise that research involving these subjects is extensive and diverse. In this section,

[2] <http://www.music.mcgill.ca/~jordan/structure>

we narrow our definition of "form" and specify what type and format of music shall be focused on in this thesis.

### *1.3.1 Definition of the word "form"*

Writing a technical-minded work on a topic as complex and enigmatic as musical form can be a treacherous endeavour, since, unlike relatively well-defined concepts such as "pitch" or "meter," "the definition of the word 'form' has been the subject of aesthetic debate for centuries" (Arnold et al. 2010). In addition, the most useful definition of "form" can vary significantly according to the musical genre under consideration; or, as Arnold et al. continue: "in musical context 'form' cannot be separated from content." Those studying music composed in the Romantic era may thus have completely different concerns from those studying 21st-century popular music—or those studying medieval masses, for that matter.

Complicating this is the fact that within the MIR community, the concept known as "musical form" to music theorists is usually referred to as "musical structure." This can cause some confusion since the general term "structure" has been used to describe topics as diverse as melodic phrasing, patterns of rhythms or note onsets, or even between-song similarities. As a result, much research published on the topic of "musical structure" has little to do with the restricted notion of structure as form that will be considered in this work.

Specifically, we will restrict the meaning of "structure" to mean how a piece is constructed out of distinct, large-scale sections, which may repeat several times. Analyzing structure thus means discovering what these sections are, and providing them with labels to indicate which sections are similar to or repetitions of each other. This is a significantly more straightforward understanding of form than those invoked in Schenkerian analysis or in the Generative Theory of Tonal Music (Lerdahl and Jackendoff 1983), two analytical approaches that yield information about tonal evolution and hierarchical grouping structure, respectively. By directing our attention to large-scale sections, this definition of form also distinguishes the current topic from the related topics of pattern recognition and melodic segmentation, which both usually focus on small-scale structure.

### *1.3.2 Types of music under consideration*

This simpler notion of structure is valid for many familiar types of music, including verse-chorus form, which consists mainly of alternating repetitions of two prominent sections, the verse and the chorus. But it applies just as well to any other primarily sectional form: for instance, many blues pieces are constructed out of identical repeating 12-bar harmonic progressions; dance-form pieces with a ternary structure and strict repetitions were common in the Classical era; and constructing larger pieces out of strictly repeating sections is a compositional procedure shared by many folk music traditions.

However, this notion of form is perhaps too simple for many other types of music: to divide a fugue into sections with discrete boundaries may be difficult, and an analysis of a sonata that did not describe the tonal relationships between sections would likely be considered incomplete. Because all of the algorithms being considered here are guided by this simpler notion of form as a decomposition into sections, we divert our attention from those musics for which this is *ab initio* a weak analytical approach, so as not to stack the odds against the algorithms. We avoid works whose primary organizational scheme is developmental, motivic, or contrapuntal—in short, works that are through composed—and instead consider works of the kind described in the previous paragraph.

### *1.3.3 Data formats under consideration*

An additional restriction to the scope of this work is that we mainly consider automated methods of structural analysis that operate on an audio signal, rather than on a symbolic representation of the music such as MIDI (Musical Instrument Digital Interface). A MIDI file encodes high-level information about a piece's pitch, rhythm, and instrumentation, and therefore could be the preferred format from which to begin an analysis. However, as a pragmatic concern, such files are rarely available for popular music, and so their study does not feature prominently in the literature reviewed in the next chapter.

Having defined and narrowed the scope of this research, it should be mentioned that there exists a lively parallel field of research, in which specific types of structural descriptions are sought for through-composed classical works, usually in symbolic

format. For instance, efforts to automate GTTM analysis are practically as old as the theory itself, given that it is already explicitly rule-based. (See Hirata et al. 2007 for a detailed review.) Efforts to automate Schenkerian analysis are similarly long-standing (see Smoliar 1980; Marsden 2007; Kirlin 2009). Cambouropoulos' (1998) *General Computational Theory of Musical Structure* describes a system for segmenting melodies that draws from several theorists' work, including Nattiez's work on paradigmatic analysis and Narmour's work on the perception of melody. Since they address symbolic representations of music only, all of this work falls outside the scope of this thesis.

### *1.3.4 Summary*

In summary, the focus of this thesis is on automatic structure analysis algorithms which: (1) treat structure as the large-scale organization of the piece into distinct, labelled sections; (2) consider genres of music for which it is valid to presume that this type of structure exists, most commonly verse-chorus and song-form music; (3) operate on audio signals, rather than symbolic score-like representations.

## 1.4 Organization of the present work

This thesis is organized as follows. The next chapter is a literature review that summarizes previous approaches to automatic music structure analysis. In Chapter 3, the implementation of the three structure analysis and two segmentation algorithms that are compared in this work is discussed in greater detail.

Chapter 4 addresses issues pertaining to the annotation and evaluation procedures. It describes the two corpora used in this work that were developed previously, as well as the new corpus of jazz and classical pieces that was collected and annotated for this work. Chapter 4 also explains the many measures used to evaluate the performance of structure analysis algorithms. The results of this evaluation are presented and discussed at length in Chapter 5, including comparisons across different corpora, algorithms, and other parameters. The work is concluded in Chapter 6.

# 2 Literature Review

In the previous chapter, we defined a "structural analysis" of a piece of music as its decomposition into large-scale sections, combined with a set of labels for these sections indicating which are similar to or repetitions of each other. The present chapter gives an overview of the methods that have been used to produce such analyses algorithmically.

Although the focus of this thesis is on structural analysis as defined in the previous chapter, this literature review includes discussion of some related sub-problems, described in Section 2.1. Although structural analysis methods differ greatly with respect to the techniques used, they all begin by extracting features from the audio, a step described in Section 2.2. The structure analysis methods themselves are discussed in Section 2.3. In Section 2.4, a variety of practical applications for structural analysis are outlined.

## 2.1 Scope of related sub-problems

As stated previously, the goal of structural analysis is to discover a piece of music's large-scale organization into distinct, labelled sections. But while the focus of this literature review remains on structural analysis, many of the methods summarized here do not all have a complete structural analysis as their final goal. A variety of related sub-problems have been addressed, including:

- **segmentation**, the estimation of a piece's structural boundaries;
- **chorus detection**, the extraction of only the most-repeated section of a piece[3];
- **repetition detection**, the extraction of all repeated segments of a piece;
- **full structure analysis**, the segmentation of a piece and the grouping of all segments that belong together;
- **semantic structure analysis**, a full analysis in which the segment groups are provided meaningful labels such as "chorus" and "verse."

[3] Note also that the application of music "thumbnailing," or producing short audio summaries of music, is strongly related to chorus extraction, and the two tasks are often described interchangeably. See Section 2.4.1.

For each of the above tasks, an example ground truth description is shown in Figure 2.1 for the song “Yesterday” by The Beatles. (“Ground truth” is generous wording for something that cannot actually be estimated with objective certainty; this issue is discussed more extensively in Section 4.1.) The information in each kind of analysis is different, but strongly related. This work focuses primarily on systems that produce “full structural descriptions,” without necessarily providing semantic labels. However, oftentimes the output of a “repetition detection” system can easily be considered a full structure analysis if each non-repeating section is simply assigned to a unique group. Also, some algorithms that only estimate a segmentation are also evaluated in this work.

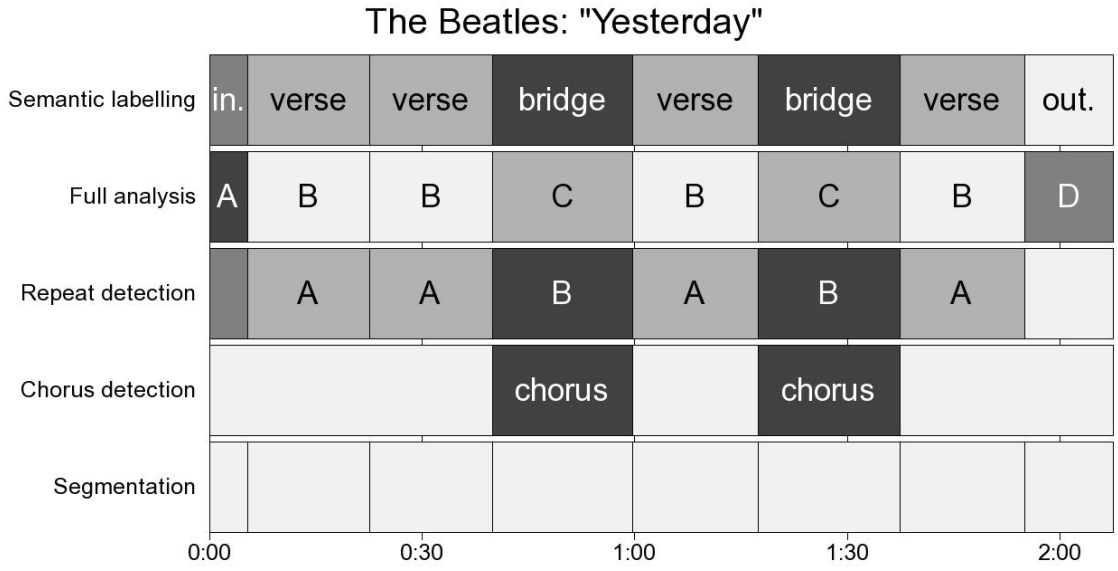



Figure 2.1. Example of ground truth annotations for the song “Yesterday” by The Beatles for the purposes of five different sub-problems in music structure analysis.

## 2.2 Feature extraction

As stated in Section 1.3.3, the methods for automatic structure analysis discussed here operate on audio data. Since audio data is very large and frequently noisy, we rarely work with it directly, and instead we extract features. This step is an opportunity both to reduce the dimension of the data and to isolate whatever information is considered most important for structural analysis. For each feature, the audio is partitioned into windows, and the data in each window are transformed into a more compact representation of some

musical attribute. The result is a sequence of feature frames, where each frame is represented by a vector of values.

In this section, after a brief word on pre-processing the signal, we describe the calculation of commonly used features relating to the musical parameters of pitch, timbre, and rhythm.

### *2.2.1 Pre-processing*

Before the feature extraction methods below are applied, the audio signal is divided into short, potentially overlapping frames, and a separate feature vector is extracted for each frame. These frames are usually very short, on the order of 20 milliseconds long, because feature extraction can be far less efficient on larger windows. However, this resolution may be far too high for the purposes of structural analysis, where one seeks to segment the audio into chunks on the order of 20 seconds long. On the other hand, taking the mean of a feature over several frames—that is, the mean and standard deviation of a feature.

An alternative approach to windowing is to run a beat tracking algorithm to estimate the locations of beats in a piece (e.g., Levy and Sandler 2008, and many others), and use the assumption that structural boundaries should be restricted to beat locations. The spans between beats, on the order of 250–500 milliseconds long, are taken as non-overlapping windows. Beat tracking generally works by calculating an onset detection function that estimates the likelihood in time that a note or percussion onset occurs in the audio. This function may be used to estimate the tempo of the piece and the beat locations. For a review of beat tracking, see Hainsworth (2006).

### *2.2.2 Pitch features*

The pitch of a sound is perhaps its most salient attribute in a musical context. The perceptual quality of pitch is characterized physically by estimating the frequency content of a sound. This may be done in a rather straightforward manner by computing the spectrum with the Fourier transform (FT), which estimates the power at each linearly-spaced frequency bin. The raw output of the FT was used as a feature by Foote (1999), Chai (2003) and Van Steelant et al. (2002). However, it should be noted that for the

analysis methods described in Section 2.3, it is important that audio frames that "sound the same" have similar feature vectors. Furthermore, especially for complex learning algorithms, having very large feature vectors can mean prohibitively long computation times. Thus the spectrum is usually processed further to achieve a better trade-off between concision and accuracy.

Since the perception of pitch is logarithmic, the frequency bins of the FT may be clustered together to rescale the frequency axis; a variation of the FT called the constant-Q transform (CQT) uses this scaling. Most music does not make arbitrary use of the continuous frequency axis, but confines itself to a discrete set of notes, such as the 12-tone scale used widely in Western music. Because of this, the resolution of the CQT is often set to 12 bins per octave to mimic this scaling, so that each bin corresponds to a single note. Finally, because the perception of pitch is cyclic, with notes an octave apart belonging to the same pitch class, the output of the CQT may be compressed by summing all contributions to the same pitch class. The result is a 12-bin histogram called a chromagram or harmonic pitch class profile (HPCP). Both concise and powerful, the chromagram has been shown to be very useful as a summary of the pitch content in a signal, and accordingly it is the most commonly used pitch-based feature; see, for example, Bartsch and Wakefield (2001), Goto (2003a), Shiu et al. (2006), and many others.

Some researchers have used variations on the chromagram that include slightly more detail: for instance, Paulus and Klapuri (2006) used a 36-bin chromagram, which kept separate the contributions from the low, middle, and high frequency ranges of the CQT (although this large vector was reduced to 15 dimensions using Principal Component Analysis, described later). Lu et al. (2004) used the raw output of the CQT, but only considered components within the main vocal range, again leading to a 36-dimensional vector. This approach has the advantage of preserving intervallic pitch relationships, information which is absent from the chromagram, but on the other hand it may fail to detect the similarity between two identical harmonies spaced octaves apart.

Another variation is to calculate a CQT with a greater number of bins per octave to provide higher pitch resolution: the 24-bin chromagram used by Gómez (2006) offers

quarter-tone resolution, for instance. Gómez et al. (2006) have even used a 120-bin chromagram with 10-cent resolution.

One potential drawback to the chromagram is that it does not account for the presence of harmonics in sound. Notes played on instruments consist of a fundamental frequency (corresponding to the heard pitch) as well as a number of harmonics, most of which are in a different pitch class than the fundamental. When two feature vectors are compared with each other, these chromagram components can lead one to misestimate the similarity between two frames. This can be partly mitigated by normalizing each chroma vector to have constant power.

To make chromagrams more robust, Gómez (2006) proposed not simply mapping each frequency bin *f* to the pitch class to which *f* belongs, but adding contributions (in decreasing amounts) to all the pitch classes which might have produced *f* as a harmonic, namely *f/2, f/3, f/4,* and so forth. This scheme was used for structural analysis by Ong (2007). Müller and Kurth's (2006) version of the chromagram represents another solution to this problem: assuming that each chroma vector should have relatively few important components, he quantized each bin according to their relative power: a component contributing more than 40% of the vector's power received the value 4; a component with at least half that power received 3; and so forth. Components contributing less than 5% were zeroed.

Another way to account for harmonics was used by Lu et al. (2004), although it was implemented at the similarity-estimation stage, described in Section 2.3.1. They enhanced the estimation of whether two chroma vectors were similar by computing the difference between chroma vectors. If there were prominent spikes in the difference vector separated by intervals related to strong harmonics (i.e., intervals of an octave or fifth), then the similarity estimate was boosted, since the mismatch between vectors was likely due to timbral differences. In a similar vein, Su et al. (2009) computed a "dissonance level" feature by estimating the prominence of minor second and tritone intervals in the unwrapped chromagram.

All of the pitch-based features described so far provide a vector where each component estimates the degree to which a pitch or pitch class is present in the sound. In this way they are all effective at capturing information related to harmony. However, if

one wishes to focus one's attention on the melody, one may obtain a score-like representation of the music using techniques that estimate the most prominent fundamental pitch. One such technique, used by Dannenberg and Hu (2002) and Chai and Vercoe (2003) is based on the autocorrelation function. Each value of the autocorrelation function estimates how well the signal matches up with a version of itself delayed by a particular duration. The best delay, selected with a peak-detection algorithm and some heuristics, will correspond to the fundamental pitch. Other methods have been developed for fundamental pitch estimation; de Cheveigné (2006) provides a good summary of this field.

### *2.2.3 Timbre features*

Timbre may be defined negatively as that attribute of sound that corresponds neither to pitch, nor loudness, nor duration. Colloquially, it is what allows one to distinguish the same note played on two different instruments. Physically, timbre is usually characterized by describing the envelope of a sound's frequency spectrum. Several attributes of the spectrum, such as its centroid, roll-off, and flux, may be taken as single scalar values, as did Xu et al. (2005) and Ong (2007). More frequently, a feature vector that describes the shape of the spectrum is used.

Mel-frequency cepstrum coefficients (MFCCs) are perhaps the most commonly used timbral feature, and are used by nearly half of the studies summarized in this chapter. Originally proposed by language researchers for performing speech analysis (Mermelstein 1976), they characterize periodicities in the spectral envelope of a sound and are effective at discriminating timbres. Upon dividing the audio signal into windows, the procedure for calculating the Mel-frequency cepstrum is as follows:

- Obtain the spectrum of the sound using the Fourier transform.
- Convert this signal to the Mel scale, which defines frequencies that are evenly spaced according to human perception. This signal is usually reduced by summing components into 40 equally spaced bands.[4]

[4] Steps #1 and #2 may equivalently calculated by passing the audio signal through a special Mel-scaled filter bank with 40 filters, and arranging the output of each filter into a vector.

- Take the logarithm of the result. This essentially decouples the sound's timbral information from its pitch.
- Calculate the discrete cosine transform of the result to produce a signal representing the periodicities in the log-spectrum. These values represent the Mel-frequency cepstrum.

Usually only the first 13 coefficients are preserved at this step, since the higher-frequency periodicities tend to correlate with the pitch of the original sound and not with its timbre. The first coefficient estimates the overall power in the signal and is also sometimes dropped after normalizing the rest of the vector. However, some have chosen more than 13 coefficients: Peiszer (2007) used all 40 coefficients, whereas Abdallah et al. (2005) reduced this set using principle component analysis (PCA) to produce a new collection of 20 maximally uncorrelated coefficients. Roughly stated, PCA detects correlation between vector values and eliminates these redundancies in the data by projecting each feature vector onto a smaller set of values. These projected values for each frame can substitute the raw spectral data in a more compact form.

Maddage et al. (2004) proposed using octave-scale cepstrum coefficients (OSCCs), a variation of MFCCs that uses a different frequency scale: rather than separate the entire spectrum into 40 perceptually-scaled bands, the bands are packed more densely near the centre of the singing range (~250–1000Hz), and less densely outside that range. They reported that OSCCs were more robust for distinguishing mixtures of instruments—specifically, distinguishing portions with singing from portions without.

Cepstral analysis seeks to characterize the spectrum by treating it as a signal itself and estimating its periodicities. However, the shape of the spectrum may be described in a more direct way by partitioning the spectrum into a set number of bands and estimating the relative power in each one. The MPEG-7 standard[5] defines its AudioSpectrumEnvelope descriptor in this way, recommending the use of 8 logarithmically-spaced bands between 62.5 Hz and 16 kHz, plus two additional bands for frequencies outside this range, although denser spacings are included in the standard (Wellhausen and Höynck 2003).

[5] A description of the MPEG-7 standard and its purpose may be found online. <http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm> accessed 10 August 2010.

Levy et al. (2007) used a related approach, also defined in the MPEG-7 standard as the AudioSpectrumProjection descriptor. For this feature, the spectral envelope is calculated in the same manner as above, but with much higher resolution: Levy et al. used bands spaced 1/8th octave apart. The very large number of values in the resulting set of feature vectors can make further processing difficult, so PCA is used to shrink them.

### *2.2.4 Rhythm features*

Although harmony and timbre are vital to one's perception of which instants sound similar to each other, our perception of music depends on these instants unfolding in time. The way that they are arranged in time, with various accents or rhythmic patterns, is vital to one's perception of which *passages* sound similar to each other. Capturing this information is the goal of rhythmic features. While much less commonly used than pitch- or timbre-based features, rhythm-based features have proven useful in some studies on structure analysis (e.g., Jensen et al. 2005; Paulus 2009). Rhythm features are usually derived from lower-level features, such as the loudness envelope, or perceptual spectral flux (PSF), which has been shown to correlate strongly with a listener's perception of note onsets.

One means of obtaining a compact view of the rhythmic content of a sound is called a rhythmogram, which estimates the prominent inter-onset intervals in the sound signal. Jensen (2005) produces a rhythmogram in two steps: first, he calculates the perceptual spectral flux (PSF) function of the signal, a feature which has been shown to correlate strongly with a listener's perception of note onsets; second, he calculates the autocorrelation function of the PSF function, to show how well the signal matches itself at different lags. This is the same technique described previously for pitch estimation, except that here a larger time scale is involved: whereas the period of the fundamental pitch may be on the order of 1 to 10 milliseconds (100–1000 Hz), Jensen's rhythmogram only considers rhythmic periods larger than 100 milliseconds (0–10 Hz).

Fluctuation patterns (FPs), described by Rauber et al. (2002), are similar to the rhythmogram, except that they use the loudness envelope instead of PSF as a mid-level feature. To this they apply the Fourier transform, detecting component frequencies instead of periods. Additionally, they calculate a separate fluctuation pattern for several

discrete frequency bands, divided according to the Bark scale, which roughly models the critical bands of the inner ear. Despite the very large size of the resulting feature vector (a 50-element FP is calculated for each of 24 frequency bands in each frame, producing 1200 values), it can be greatly reduced with PCA; Rauber et al. (2002) reported reducing it to 80 principal components with minimal loss of variance. FPs were subsequently used by Peiszer (2007) and by Su et al. (2009); Turnbull et al. (2007) used a variation with bands divided according to the Mel scale instead of the Bark scale.

In a global sense, rhythm is related to tempo, or the speed of a piece of music, expressed in beats per minute. Tempo is often deduced by detecting the most common inter-onset intervals in the signal, and is calculated as a preliminary step in the beat-tracking algorithms mentioned earlier. Kim et al. (2006) made novel use of tempo estimates as a feature: observing that different sections of music often produce different tempo estimates (since tempo estimation algorithms are prone to make octave errors, misestimating the true tempo by a factor of a power of 2), they used the frame-wise tempo estimation as an initial segmentation of the piece.

The rhythm descriptors stand apart from the pitch and timbre features because they incorporate information about how the music unfolds in time. However, one may easily augment features with timing information by calculating the instantaneous derivative between neighbouring feature vectors, as in Tzanetakis and Cook (1999) and Su et al. (2009). Higher-order derivatives may be calculated indefinitely, but the computational cost of having feature vectors with too many dimensions discourages this; it does not appear that anyone has gone beyond the second derivative. Turnbull et al. (2007) used derivative features, as well as what they termed "difference" features: these measured for each frame the dissimilarity between the average feature values for a short period before and after.

Derivatives can show the degree to and the rate at which pitch- and timbre-based features change in time. To capture information on the *frequency* with which they change, Peeters et al. (2002) used what he termed "dynamic features." These are calculated in much the same way as the fluctuation patterns described above: after producing, say, the MFCC magnitudes for each frame, he calculated the spectrum of each component's fluctuations over time using the Fourier transform. By varying the window

over which the FT is taken, one can vary the time scale being examined: a small window will pick up short-term fluctuations, and a large window will estimate long-term fluctuations. In either case, the result is a large matrix for each frame, where element (*i, j*) measures the degree to which MFCC band *i* fluctuates with frequency *j*. Again, the large size of each feature vector becomes a problem: to identify the most important matrix elements, Peeters et al. used a supervised learning technique which maximized the mutual information between feature values and the section classes defined by annotated pieces of music.

## 2.3 Approaches to structural analysis

In the previous section, a variety of mid-level representations characterizing musical features were described. What remains is to analyze this information and produce a structural analysis. The next section introduces some ideas that are important in understanding the structural analysis methods summarized in this chapter. This is followed by a review of top-down approaches and then bottom-up approaches.

### *2.3.1 The sequences hypothesis vs. the states hypothesis*

In pursuing a structural analysis algorithmically, there are two basic assumptions that one may make about how structure is reflected in music, which correspond to the two basic analytic strategies. Peeters (2004) described them as the "states" and the "sequences" approaches, and most analysis systems may be classified as taking one or the other approach. These different approaches are discussed in this section.

In the sequences approach, the algorithm searches a series of feature vectors for repeating sequences. The algorithm must then deduce from these repetitions what the structure of the piece is. If the different sections of the piece each have distinct sequences of features—for instance, if the chorus and verse have different melodies—then this approach may succeed. In the states approach, the goal is to look for sections of the piece which are homogeneous with respect to some feature, or which occupy a single "state." For instance, suppose the chorus of a song used the chords C, F and G, whereas the verse mainly used the chords A and E; each section would thus occupy a particular harmonic

state. Rather than detect a particular repetition, then, the states approach relies on estimating what the important states are, and discovering when they reoccur.

Each approach implies an assumption about how music is structured. To understand their respective advantages, it is helpful to consider some examples. In ideal circumstances, both approaches should be able to correctly analyze a song with ABA structure. However, given the structure AAB, the states approach would not recognize the boundary between the two A sections, and would produce the analysis AB. On the other hand, the structure ABC, containing no repetitions, would stymie the sequences approach. The possibility of a song with structure AABCA demonstrates the need, in theory, to consider both sequence and state information. These situations and a couple others are summarized in Table 2.1.

| *True structure:* | *Sequence analysis:* | *State analysis:* |
|---|---|---|
| ABA | ABA | ABA |
| AAB | AAB | AB |
| ABC | A | ABC |
| AABCA | AABA | ABCA |
| ABAB | AA | ABAB |
| ABABB | ABABB | ABAB |

Table 2.1. Ideal structural analysis provided by the sequences and states approaches for different true structures.

Many of the algorithms discussed in this chapter can be seen as exclusively applying either the sequence or state approach. However, several use information from both approaches, sometimes in separate stages (e.g., Cooper and Foote 2003), and sometimes all at once (e.g., Paulus and Klapuri 2008b). Although the present literature review is organized according to methodological approach, the states and sequences perspectives will be referred to frequently.

In Section 2.3.2 we discuss the top-down approach, in which an overall view of the piece is produced (generally in the form of a self-similarity matrix) and deconstructed

into separate sections. In Section 2.3.3 we discuss the bottom-up approach, in which some form of clustering algorithm builds an analysis of the piece from individual frames or short sequences.

### *2.3.2 Top-down approach: Self-similarity matrices*

The top-down approach to music structure analysis involves producing an overall view of the piece and extracting from it information on the location of boundaries between musical sections and the similarity of these sections to each other.

The overall view is usually computed as a self-similarity matrix (SSM), alternatively called a self-distance matrix[6], a method originally proposed by Jonathan Foote (1999) as a means to visualize repeating patterns in music. SSMs are very similar to recurrence plots, which Eckmann et al. (1987) proposed for better understanding the evolution of dynamical systems. As a technique for visualizing repeating patterns embedded in any long, complex sequence, recurrence plots have since been applied to many research domains, including DNA sequence analysis. Foote was the first to apply this technique to music analysis.

Given a sequence of $N$ feature vectors, the $N$x$N$ self-similarity matrix is generated by filling the pixel at $(i, j)$ with the similarity between vectors $v_i$ and $v_j$. The similarity is usually computed using either the Euclidean or cosine distance between the vectors. The Euclidean distance $d_E$ corresponds to the geometric length between two points, while the cosine distance $d_C$ finds the angle between two vectors. Considering two vectors $v_i$ and $v_j$, these distances are calculated as:

$$d_E(v_i, v_j) = \|v_i - v_j\|$$

$$d_C(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\| \cdot \|v_j\|}$$

where $\|x\|$ denotes the vector norm of $x$, i.e., its dot-product with itself:

$$\|x\| = \sum_i |x_i|^2$$

[6] There is no practical difference between a self-similarity and a self-distance matrix. In this thesis, all such matrices are referred to as SSMs, and in all visualized SSMs, similar regions show up as white and dissimilar regions as black.

The Euclidean distance function is theoretically unbounded and so is sensitive to the magnitude of the vectors. On the other hand, the cosine distance function is normalized by the magnitude of the two vectors being compared, and outputs values between 0 and 1. The use of a scalar distance measure is part of what distinguishes usage of SSMs for music analysis from earlier recurrence plots, which typically used binary output values (i.e., indicating whether elements *i* and *j* in a string were identical or not). Note that half of the information in an SSM is redundant if the distance measure is symmetric, although the full matrix is usually presented anyway.

When the features describe distribution statistics (e.g., the mean and variance of MFCCs over a particular period), one may use the Mahalanobis distance metric, which estimates the correlation between two distributions (e.g., Xu et al. 2002), or the Kullback-Leibler (KL) divergence, which estimates how much extra information would be required to describe one distribution based on a code provided by the other (e.g., Foote and Cooper 2003). The KL divergence $d_{KL}(v_i, v_j)$ is not symmetric, so a symmetrized version is usually produced by summing the distances: $d_{KL}(v_i, v_j) + d_{KL}(v_j, v_i)$. Mauch et al. (2009) used the Pearson correlation coefficient, which estimates the linear dependence between two feature vectors from their covariance.

An SSM can vividly illustrate the internal structure of the music. In particular, two aspects of structure are often clearly represented: boundaries between sections, and repetitions.

#### 2.3.2.1 Boundaries

Intuitively, we can claim that a boundary is felt in a piece of music whenever something changes. Certainly this is part of the listening experience: perception experiments by Bruderer et al. (2006) confirm, for instance, that many structural boundaries may be attributed to changes in timbre. A sudden change reveals itself in an SSM as a checkerboard pattern (see Figure 2.2a) on the main diagonal: the corners of these patterns indicate moments at which the recent past is self-similar, the near future is self-similar, but the past and the future are not similar to each other. On a small scale, these corners may indicate the boundary between two different notes: in Figure 2.3a, the SSM for the first couple seconds of a recording of "Layla" by Derek and the Dominoes, the main diagonal (from bottom left to top right) is plainly punctuated with corners

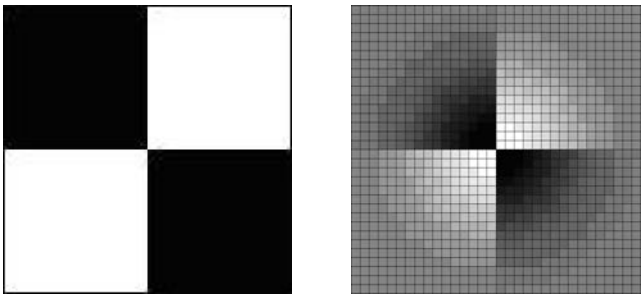


Figure 2.2a (left). Ideal checkerboard kernel.

Figure 2.2b (right). Gaussian-tapered, 32-pixel square checkerboard kernel.

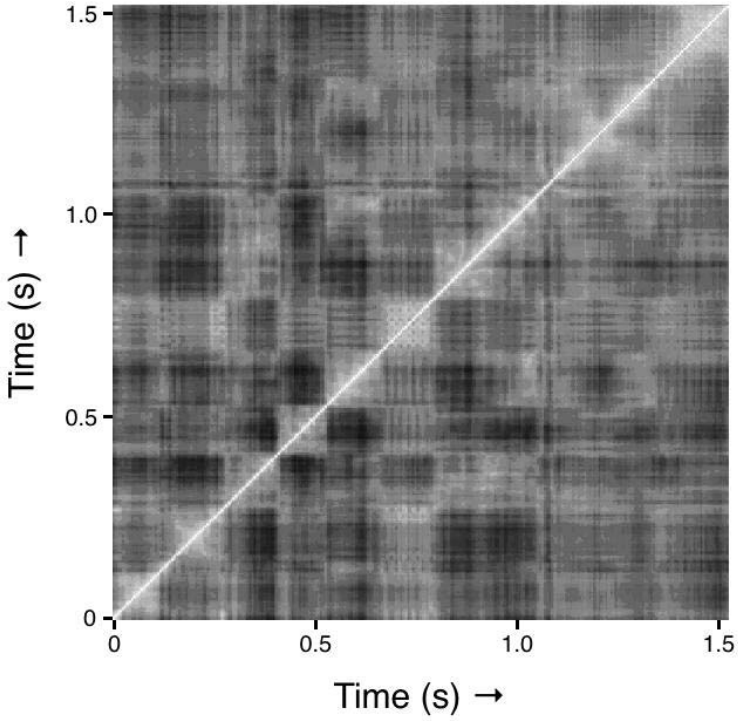



Figure 2.3a. SSM for the first 1.5 seconds of a recording of "Layla" by Derek and the Dominoes using Fourier transform coefficients as features. Time proceeds linearly from left to right and from bottom to top. The pixel $(i, j)$ therefore refers to the $i^{th}$ pixel from the left and the $j^{th}$ pixel from the bottom. The six short notes of the opening riff (see Figure 2.4) are visible as white squares along the main diagonal, and the sustained note that follows is visible as a relatively homogeneous white region in the top right of the SSM.

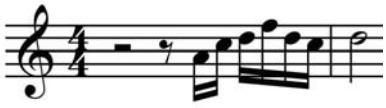


Figure 2.3b. Corresponding score for the sound depicted in Figure 2.3. Score obtained from Wikipedia, <http://en.wikipedia.org/wiki/Layla> accessed 4 August 2010.

marking the onset of each new note (a score transcription of the excerpt is in Figure 2.3b). Each of these corners indicates an instant where the previous short time span and the time span to follow are both internally homogeneous, since they represent periods of near-constant pitch, but where these time spans are very dissimilar to each other, i.e., they are different pitches.

This same pattern may be seen at larger time scales. Figure 2.4 shows the SSM for the entire length of the same recording of "Layla" using MFCCs as features. That song famously consists of two starkly different halves: the first is a familiar alternation of verses and choruses, ending in a guitar solo; the second part, beginning at around 3:10, is an extended instrumental section that includes a prominent piano part. Since the two halves are in different keys, the same turning point is also readily visible in Figure 2.5, an SSM for "Layla" using chroma vectors as features. This image also reveals a second moment of great change at around 2:20; indeed, this is when the repeating melodies of the verses and choruses end and the non-repeating guitar solo section begins. The difference between the two SSMs illustrates the fact that different features often capture different aspects of musical structure.

To detect these moments of change automatically, Foote (2000b) proposed correlating each point along the main diagonal with a checkerboard kernel tapered with a Gaussian function (Figure 2.2b). When the kernel is centered on a moment of change, the correlation will be high, corresponding to a peak in the resulting novelty function. By smoothing the function and using an appropriate peak-picking function, the prominent boundaries of the piece can be estimated.

The checkerboard kernel is usually Gaussian tapered to be less sensitive to noise, and its size may be adjusted to pick out structural boundaries of a particular scale. Figure 2.6 shows two novelty functions produced for the same song, "Layla," but at different temporal scales. Both kernels detect the important midway point near 3:10, but its relative significance is seen best when measured with the larger kernel, illustrating its sensitivity to larger-scale structure.

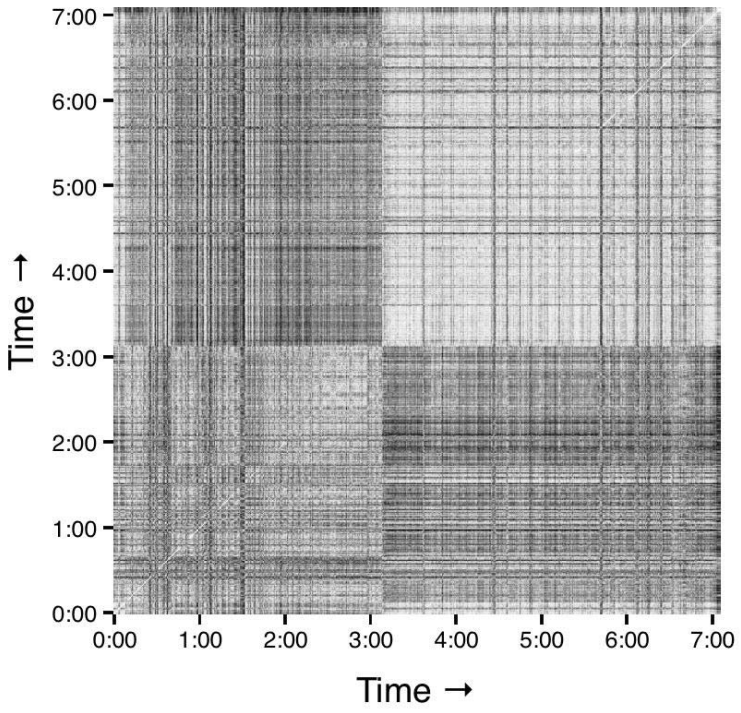



Figure 2.4. SSM for the entirety of the song "Layla" using MFCCs as features.

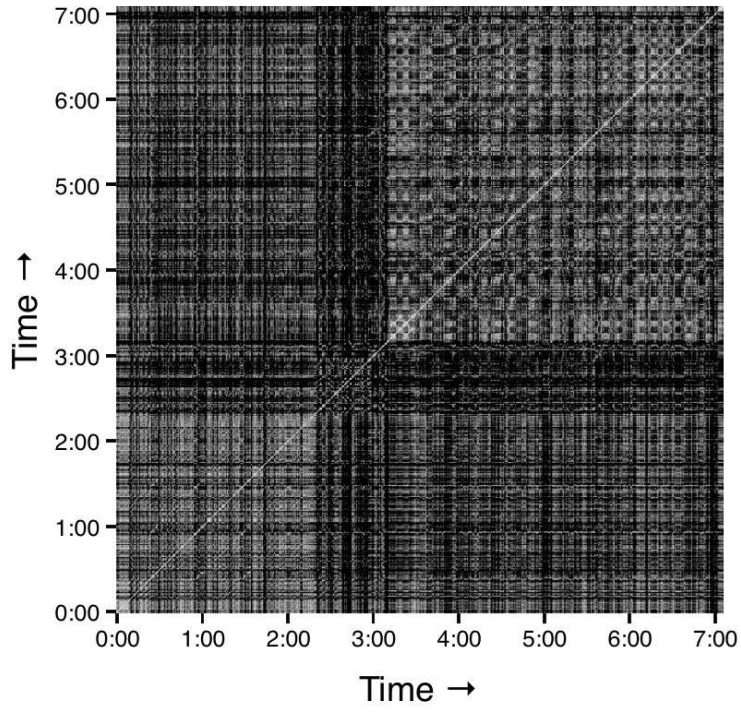



Figure 2.5. SSM for the entirety of the song "Layla" using chroma vectors as features.

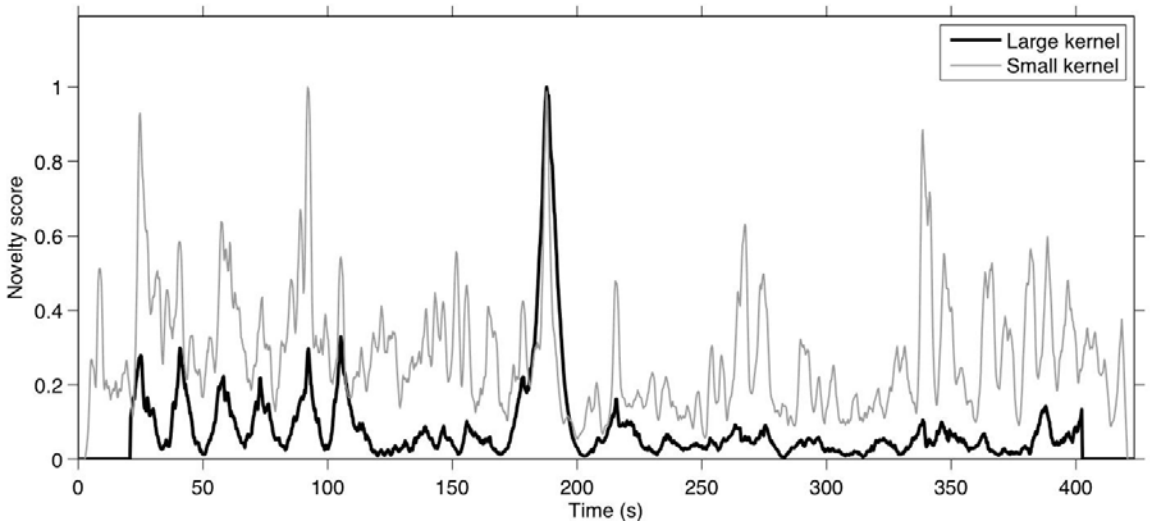



Figure 2.6. Two novelty functions generated by convolving a Gaussian-tapered checkerboard kernel along the diagonal of the chroma-based SSM shown in Figure 2.5. The small kernel had a width equivalent to 2% of the length of the song, while the large kernel had a width equivalent to 10% of the length of the song.

#### 2.3.2.2 Repetitions

Self-similarity matrices also reveal when material is repeated in a piece, which is very important to know when analyzing the kind of sectional music under consideration here (discussed in Section 2.1.2). As Peeters (2004) points out, repeated sections may be reflected in SSMs in one of two ways: as stripes parallel to the main diagonal, indicating reoccurring "sequences" of music, or as blocks, indicating reoccurring, internally homogeneous "states." The presence of either feature depends on the type of music being considered and the feature used to generate the SSM, so each are described in turn.

Consider a diagonal line segment $n$ seconds long that is parallel to an SSM's main diagonal, and which begins at the point $(i, j)$. It indicates that the $n$-second sequences ($i$, $i+1$, $i+2$, ... $i+n$) and ($j$, $j+1$, $j+2$, ... $j+n$) are similar to each other; that is, the material beginning at time $i$ repeats at time $j$. Such line segments may be observed in Figure 2.7a, an SSM based on chroma vectors for the song "Yesterday" by The Beatles. A picture of the ideal diagonal line segments anticipated for that song, based on its structure as annotated by a human listener, is displayed in Figure 2.7b. The sequences in this song are prominent because while none of the sections are harmonically static, the melodies and harmonic progressions of each part repeat closely.

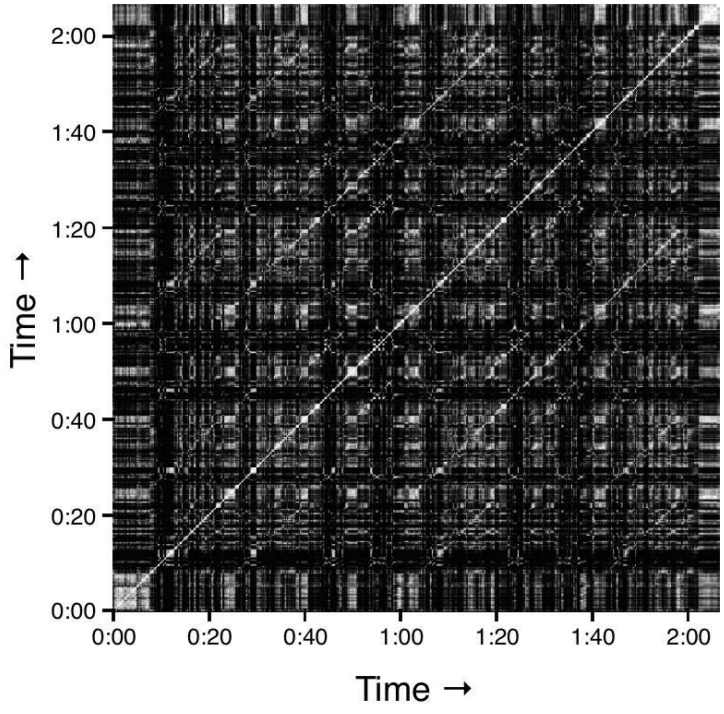



Figure 2.7a. SSM for the song "Yesterday" using chroma vectors.

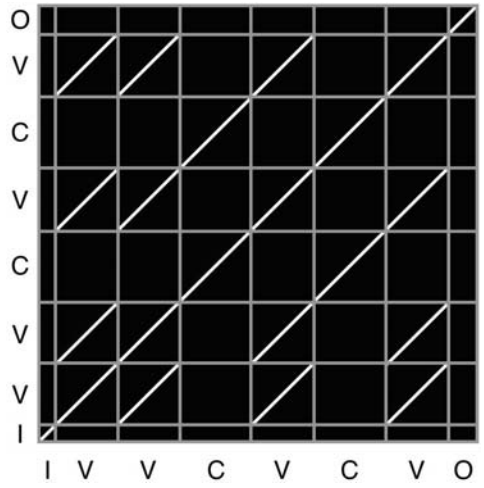



Figure 2.7b. Ideal sequence representation for SSM of "Yesterday." The letters I, V, C and O stand for "intro," "verse," "chorus" and "outro," respectively. Gray lines indicate annotated boundaries and diagonal white lines indicate repetitions.

In other songs, sections marked by distinct timbral textures or different harmonic areas may reveal themselves as blocks instead of lines. Before, it was seen that blocks on the main diagonal indicate sections of homogeneous sound: for example, the two halves of "Layla" were each homogeneous (drawn from a single palette of chords) and different from the other (they occupied different key areas). When a homogeneous section repeats after some intervening material, a block will appear off the main diagonal. This may be observed in Figure 2.8a, the SSM based on chroma vectors for "Lucy in the Sky with Diamonds" by The Beatles. In this song, the chorus and verse occupy very different tonal spaces, so the blocks are quite clear, despite some cross-similarity between the bridge and chorus. The ideal block structure anticipated from the annotation is pictured in Figure 2.8b.

The choice of feature has an impact on the presence of lines or blocks in the SSM: the "Layla" example showed how timbral and pitch-based features can emphasize different sets of important boundaries (see Figures 2.4 and 2.5), while the two Beatles examples show that the same feature can emphasize different kinds of musical structure, depending on the properties of the song in question (see Figures 2.7a and 2.8a). Fortunately, in many pieces, different sections often exhibit simultaneous changes in timbre, harmony and rhythm, and one is likely to be able to uncover blocks or diagonal line segments in any SSM. For instance, in Figure 2.8a, the repeating bridges and choruses produce prominent blocks, while the repeating verses leave diagonal lines.

To simplify the process of filtering the matrix and enhancing repetition lines (addressed in the following section), an SSM is commonly transformed into a time-lag matrix, where each pixel $(i, j)$ encodes the similarity between frames $i$ and $i - j$. Figure 2.9a shows the time-lag SSM for the song "Yesterday" using chroma features, with the ideal sequence structure in Figure 2.9b. These contain the same information as Figures 2.7a and 2.7b, but note how the lines of constant lag in the original SSM—i.e., the diagonals—are converted into straight horizontal lines in the time-lag matrix.

Clearly, knowledge of boundaries alone is not sufficient to infer all structure: after segmentation, one needs to assess the similarity of the segments. However, repetition detection alone provides insufficient information too: some pieces have non-repeating sections that are nevertheless distinct from each other, such as the Beatles' "Happiness Is

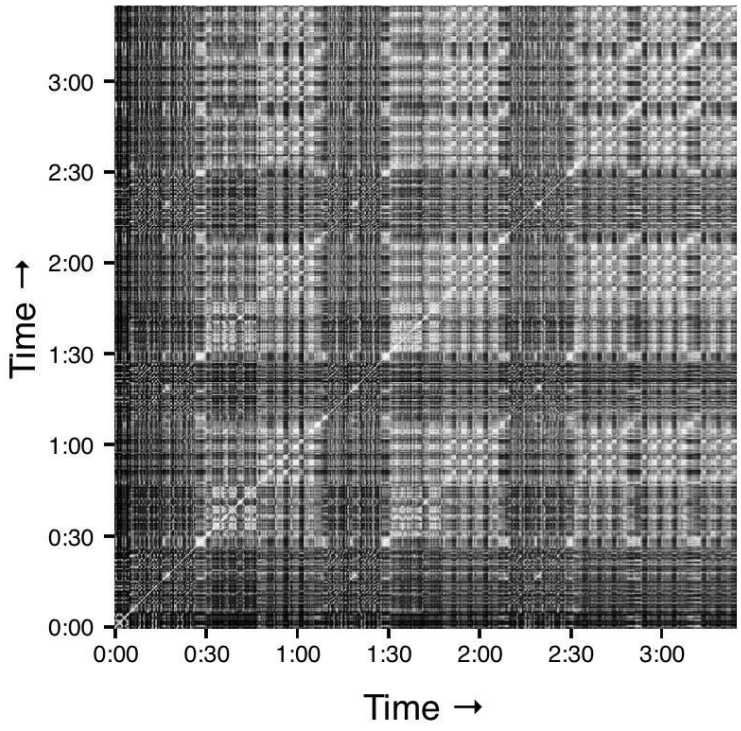



Figure 2.8a. SSM for the song "Lucy in the Sky with Diamonds" using chroma vectors.

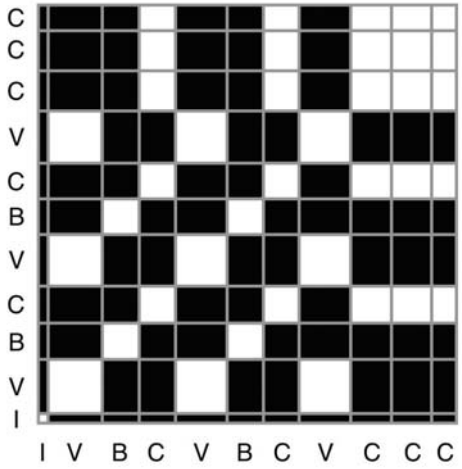



Figure 2.8b. Ideal state representation for SSM of "Lucy in the Sky with Diamonds." The letters I, V, B, C and O stand for "intro," "verse," "bridge," "chorus" and "outro," respectively. Gray lines indicate annotated boundaries and white blocks indicate repetitions.

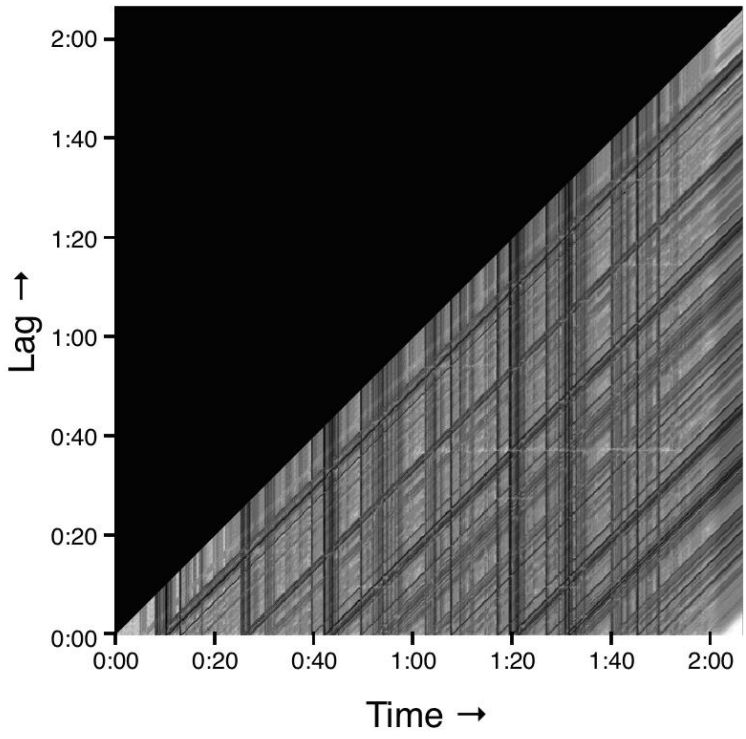



Figure 2.9a.Time-lag SSM of the song "Yesterday" using chroma vectors.

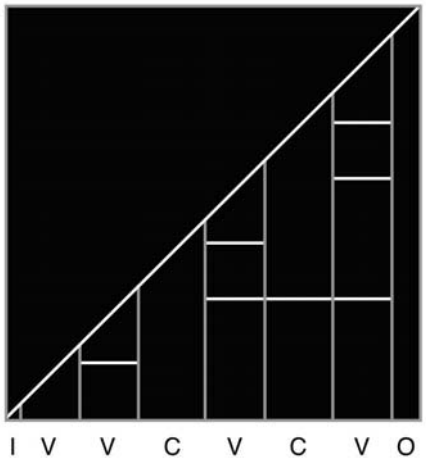



Figure 2.9b. Ideal sequence representation of time-lag SSM of "Yesterday." The letters I, V, C and O stand for "intro," "verse," "chorus" and "outro," respectively. Vertical gray lines indicate annotated boundaries and horizontal white lines indicate repetitions.

A Warm Gun," which has overall structure ABCD. The challenge in the top-down method lies in mediating these two important sources of information—points of novelty and knowledge of similar sequences or states—to construct a structural description.

#### 2.3.2.3 Manipulating the SSM

The SSM is very tantalizing: with the human eye one can easily detect the lines and blocks, and from them, with just a few simple constraining assumptions and knowledge of musical structure, one may deduce an analysis. In practice, unfortunately, directing or training a computer to perform these steps is very difficult. Researchers have experimented with countless filtering techniques and borrowed many ideas from the field of image processing to accomplish this task. These efforts are summarized here in this section.

*Image processing*

In its initial state, the SSM is often too large to deal with easily, and the lines and blocks indicating repetitions may be obscured by noise. In addition, repetitions may not be exact, and the lines which indicate them may have small gaps. These issues may be partly addressed by low-pass filtering the SSM in the diagonal direction, which emphasizes lines, and by downsampling the matrix.[7] Each pixel (*i, j*) in the resulting SSM represents the average similarity between the two sequences of equal length beginning at points *i* and *j*.

Foote (1999) did this with a running-average filter, replacing each pixel by the sum of a set of *w* pixels on the same diagonal line, and then downsampling by the same factor *w*, on the order of 5 or 10. Some, including Zhang and Samadini (2007) and Paulus and Klapuri (2008a), have chosen to low-pass filter the features themselves, smearing them in time before generating the SSM. Müller and Kurth (2006) used both approaches: their chroma vectors were low-pass filtered and downsampled in time, and further low-pass filtering was applied to the SSM in the diagonal direction. Noting that relatively silent portions of the audio could introduce noise to the SSM (in particular if the Euclidean distance is used as a similarity measure), Marolt (2006) added a small amount

[7] Downsampling is usually preceded by low-pass filtering to avoid distorting the data.

of random noise to the feature vectors to reduce spurious matches. The length of the diagonal filter applied affects the time scale over which matching melodies may be sought. Marolt (2006) calculated three SSMs using different filter lengths and combined them by element-wise multiplication to achieve a striking reduction in noise.

Working with a time-lag SSM, Goto (2006a) used an elaborate local-mean estimation technique to reduce noise. For each point, he first calculated the maximum and minimum local mean values in the horizontal, vertical, and diagonal directions. If the maximum among these belonged to the horizontal neighbourhood, then the point was considered part of a repetition line and emphasized by only subtracting the minimum local mean. Otherwise, it was considered noise and suppressed by subtracting the maximum local mean. In a later repetition line extraction stage, Goto also high-pass filtered the SSM in the vertical direction, further enhancing the sequences.

The choice of features used to generate an SSM and the filtering applied to it will affect how strongly the state or sequence structure appears in it: timbral features probably reflect states in the music, while pitch features are perhaps more apt to reflect sequences. Recognizing this, Eronen (2007) created two SSMs from MFCC and chroma features, applying Goto's sequence-enhancing directional-mean filtering technique only to the chroma SSM. The two SSMs were then summed.

Lu et al. (2004) and Ong (2005) both borrowed from the field of image processing a set of techniques known as erosion and dilation, which are used to smooth out isolated pixels while retaining edges. Erosion works by replacing each pixel with the minimum value that occurs in a defined neighbourhood around it; dilation does the opposite, replacing each pixel with its neighbourhood maximum. Applying these operations in turn will reduce the impact of isolated minima and maxima, enhancing the appearance of blocks in an SSM. The operation can be applied in one dimension, for instance, to enhance repetition lines in a time-lag SSM (Lu et al. 2004), or in two dimensions, to enhance an SSM's block structure (Ong 2005). Wellhausen and Höynck (2003) shrank their SSM and performed erosion at the same time, subdividing the SSM into square blocks and replacing each with its maximum value. To reduce noise and enhance diagonal lines, each pixel was divided by the mean of a square region around it, and multiplied by the mean along the region's diagonal.

Finally, Peeters (2007) proposed making "higher order" SSMs that take into account the transitive relationships between frames. An important problem with SSMs is that although frame *k* may be detected as similar to frames *i* and *j*, these two frames may not be detected as similar to each other. Peeters applies a transitivity operation to the SSM which boosts the value at (*i, j*) if pixels (*k, i*) and (*k, j*) have similar values; effectively, each pixel (*i, j*) is replaced with the correlation between row *i* and row *j*. A still higher-order SSM could take into account twice-removed transitive relationships. The technique appeared to greatly enhance the SSM's repetition lines.

*Beat indexing*

The size of the matrix can be reduced mechanically by downsampling, as above, but an alternative approach is to segment the audio in a musically meaningful way: this way one may use larger window sizes without worrying about obscuring detail. For instance, one may use a beat-tracking algorithm to estimate beat locations in a piece and produce a beat-indexed SSM; this was already mentioned in Section 2.1.1, and was used by Levy and Sandler (2008), Eronen (2007), Marolt (2006), Bartsch and Wakefield (2001), and others. Going a step further, Shiu et al. (2006) assumed that each measure is four beats long, and compared four-beat sequences to create a measure-indexed SSM.

Jehan (2005b) presented a method of constructing hierarchically-related SSMs that could be applied to structure analysis. Each SSM is indexed at a different time scale: starting with a standard frame-level SSM, the second SSM is indexed by short segments determined by note onsets, the third by beat, and the fourth by short, automatically detected patterns. The criteria used at each level differ as well: the frame-indexed SSM is built with timbral features; the second is produced by comparing the segments' envelopes; pitch is included as a feature only at the beat level. The length of prominent, short, repeated patterns is estimated from the beat-level SSM, and beats are merged into groups at this scale; the dynamic programming distance between each group of beats is used to fill the pattern-level SSM.

*Boosting with string matching*

Lines of repetition are unlikely to exist as perfectly straight lines at 45 degrees in the SSM (perfectly horizontal in the time-lag version), due to natural timing deviations in performance, as well as the arbitrary window size. This can be partly accommodated by

adding a Gaussian-tapered width to the diagonal filter, the approach chosen by Aucouturier (2001). However, this can blur the SSM significantly. Chai (2005) proposed using dynamic programming (DP) sequence alignment to reduce the size of the SSM and detect partial matches more robustly. DP sequence alignment, often referred to as dynamic time warping (DTW), is a technique that estimates the minimum edit distance between two strings. Chai considered a short string of feature frames beginning at frame *i* and exhaustively computed the DTW distance between it and the rest of the piece at every future frame *j*; the minima of this function indicated repetitions and were used to sparsely populate a time-lag SSM. A similarly sparse SSM was effectively produced by Zhang and Samadani (2007), although they did not use DTW.

Maddage et al. (2004) used the same approach, but incorporated automatically estimated measure information to reduce the number of alignments that needed to be computed. Shiu et al. (2006) produced an SSM indexed at the half-beat level; assuming a 4/4 time signature, this SSM was then subdivided into 8x8-pixel subregions representing measures. The DTW distance between each pair of measures was used to populate a smaller, segment-indexed SSM. Peiszer (2007) and Paulus and Klapuri (2008a) would later use the DTW distance to estimate the similarity between large-scale sections.

Dannenberg and Hu (2002), working with a rough symbolic score that had been automatically created with fundamental pitch estimation, created an SSM with built-in sequence information: in it, the pixel (*i, j*) indicates the length, in seconds, of the longest match between the substrings beginning at notes *i* and *j*, found using a heuristic string-matching algorithm. However, this was then treated more as a record-keeping device than as a visual representation: their subsequent analysis approach involved deleting redundant pixels until a very small number remained, each one indicating a repeating pair of segments.

Müller and Kurth (2006) presented a method to emphasize repetition lines even in the face of gross tempo deviations, which are possibly more common in classical music repertoire than in modern popular music. Recall that with diagonal filtering, each pixel (*i, j*) represented the average similarity between the two sequences of equal length beginning at points *i* and *j*. To make this filtering robust to variations in tempo within a single piece, Müller and Kurth also compared the sequence beginning at point *i* to a variety of tempo-

shifted versions of the sequence beginning at point *j*; the pixel (*i, j*) was taken as the maximum found similarity value.

*Modulation detection*

While tempo deviations are probably rare in popular music, modulations are relatively common: it is a trope in popular music to modulate near the end of the song to add dramatic emphasis. Using chroma vectors as features, Goto (2003a) could account for such cases by generating 12 different modulated SSMs: in each one, the pixel (*i, j*) indicates the similarity between the original chroma vector at moment *i* with the chroma vector at moment *j* pitch-shifted by some number of semitones between 0 and 11. The resulting SSMs, which are no longer symmetric, are searched for repetition lines that match segments extracted from the unmodulated SSM.

This procedure was later adopted by Ong (2007). To make analysis more straightforward, Müller and Clausen (2007) condensed the information in Goto's 12 modulated SSMs by filling the pixel (*i, j*) with its maximum value (indicating highest similarity) over the 12 SSMs. The originating SSM of each pixel is remembered to ensure that the detected repetition lines are not generated spuriously from several SSMs, and occupy a non-modulating portion of the piece.

Two other options were investigated by Chai (2005): in one case, she estimated the key of each region of the piece, and used this knowledge to transpose each region's feature vector to the home key, creating a key-adjusted SSM. She also created an interval-based SSM from chroma derivative features.

#### 2.3.2.4 Interpreting the SSM

Having emphasized the state or sequence structure of the SSM, these elements must be detected in the SSM. This process is rarely straightforward, and a variety of approaches to this problem have been pursued.

*Estimating states*

Foote and Cooper (2003) used a large checkerboard kernel to calculate a novelty function from the SSM, the peaks of which were taken as a final segmentation for the piece. Supposing *N* segments were obtained, a new *N*x*N* SSM was calculated to display the similarity between pairs of entire segments. Adhering to a states interpretation, they

treated the segments as probability distributions and estimated the symmetrized Kullback-Leibler (KL) divergence between them to populate the segment-indexed SSM. Singular-value decomposition (SVD) was then applied to cluster the *N* segments into a pre-set number of groups, resulting in a full structural description. Put briefly, SVD attempts to decompose the SSM into a small number of column vectors, and is effective in this application since all columns corresponding to repetitions of the same segment should ideally be equal.

Peiszer (2007) also used novelty functions to create an initial segmentation, and experimented with several *k*-means clustering algorithms to obtain an analysis. In this clustering technique, a set of *k* group centroids are estimated, with each data point being assigned to that cluster with the nearest centroid. The algorithm accepts a set number *k* of centroids as inputs (which may be random), and iteratively adjusts them to reduce the distance between each cluster's centroid and the data points that belong to it. Peiszer's experiments included: applying *k*-means clustering to the means of each segment's feature vectors; applying *k*-means clustering to the feature frames and then assigning segments to clusters with a voting scheme; and estimating the DTW alignment cost between segments, and incorporating that information into a *k*-means clustering scheme. Because *k*-means is highly sensitive to the initialization of the centroids, Peiszer executed each algorithm several times, keeping the results with the tightest clusters.

*Detecting repetitions*

Goto (2003a), having emphasized the repetition lines in his SSM, searched for these through several applications of an automatic threshold technique devised by Otsu (1979), which chooses a threshold by maximizing the between-class difference of the elements below and above the threshold. Goto found the sum of each row in the time-lag SSM and used Otsu's threshold method to select the rows that contained repetition lines. Each row was smoothed and its repetition lines were extracted by applying Otsu's method again. The result was a set of horizontal line segments, each specifying two regions of a song that are similar.

Consolidating these segments into related groups is a difficult task, since the detected boundaries of the repetition lines are inexact and may not reflect the true structure exactly. Furthermore, transitive relationships are not always consistently

detected: that is, if segment B is a repeat of A, and C is a repeat of B, then C is also a repeat of A and three repetition lines should be detected, but one is often missed. To overcome these hurdles, Goto developed an elaborate processing chain, including some heuristically-set parameters defined in Goto (2006a): related segments were first grouped together if their beginning and end times disagreed by less than 20% of their length, or a maximum of 3.6 seconds; a search for those repetition lines that were transitively inferred but not detected was conducted based on the detected segment groups; equally-spaced segments that repeated too often, and overlapping segments, were removed; and finally, modulated SSMs were consulted to detect transposed repetitions. Goto dubbed his system RefraiD, and its general framework has been adopted and modified by many other researchers.

For instance, Lu et al. (2004) used a similar approach, with some simplifications. Rather than detect the best rows in the SSM, they used Otsu's method to set a threshold and binarized the entire SSM. To avoid too many spurious segments, they also greatly restricted the range of the automatically set threshold to retain only 0.2–1% of the pixels, and applied erosion and dilation operations to preserve only segments with a minimum length of 16–32 beats. The remaining segments were iteratively grouped together, providing a fairly accurate structural analysis. Finally, the boundaries were refined using an optimization algorithm that adjusted the boundaries to maximize within-group similarity and to force repeated segments to have the same size.

Eronen (2007) combined elements of both approaches: Otsu's method was used to choose diagonals above a dynamic threshold (although he retained at least the best 10), the best 20% of points in each diagonal were retained, and a dilation operation was used to fill in the gaps. The subsequent analysis worked much the same as in RefraiD.

Ong et al. (2006) also followed the RefraiD approach, with a novel addition: after consolidating segments into groups, they correlated the earliest occurring segment in each group with the rest of the song to locate additional repetitions that may have been missed. All peaks within a fixed threshold of the maximum peak were taken to indicate repetitions. Ong (2007) extended this algorithm to fine-tune the boundaries (the adjustments were less than 2 seconds), using candidate boundaries produced according to Ong (2005). In the latter method, novelty functions were calculated from several SSMs

created using various features and filtered to emphasize their block structure. In this way, Ong combined both the sequences approach (to locate repetitions) and the states approach (to fine-tune boundaries).

Peeters' (2007) replaced the segment consolidation phase of the RefraiD method with a new algorithm that searched among the detected segments to establish which "mother" segments best explain each detected set of repetitions.

Zhu et al. (2005) incorporated data from karaoke videos into their method, but the SSM interpretation step was nevertheless very similar to Goto's method. An SSM based on chroma features was analyzed using RefraiD to locate repetitions in the music, and an SSM based on lyrics data[8] was analyzed to locate repetitions in the lyrics. The detected repetitions from each feature were combined afterwards.

Chai (2005), working from a very sparse SSM, took a simpler approach: she iteratively assembled segments by beginning with the most-frequently repeating segments (i.e., the columns in the time-lag SSM with the greatest number of pixels) and extending them according to the lines in the SSM. Müller and Kurth's (2007) line detection method involved a greedy, iterative path generation procedure that first used a fixed threshold to ignore most of the SSM, and then traced the optimal paths among the remaining pixels. In their algorithm, once a strong path was found, its neighbourhood was removed from consideration, to avoid detecting redundant segments. A structure analysis algorithm then grouped transitively related segments and discarded unnecessary ones.

*Hybrid approaches*

Shiu et al.'s (2006) approach took advantage of both states and sequences information. First, to obtain repetition lines, they interpreted the pixel magnitudes as transition probabilities in a path through the SSM, and then applied the Viterbi algorithm to uncover the likeliest path through the matrix. The algorithm was able to locate diagonal paths where they occurred, while being robust to inexact repetitions and slight timing deviations. To help steer the Viterbi in the correct direction, an additional

[8] As a feature, Zhu et al. simply extracted a small image of each character in the lyrics that appear in the video, and estimated two characters as being the same if more than 75% of the pixels in each image matched up. By avoiding the complexities of character recognition, they produced a system that was both language- and script-independent.

constraint was applied, based on a states interpretation of the piece: the main diagonal of the SSM was correlated with kernels designed to detect a tendency to immediately repeat every 2 or 4 measures. If a strong correlation was detected, a region was tagged with an appropriate label. The similarity between any two regions that had a different label were effectively zeroed, and transitions between them ignored in the Viterbi step.

Paulus and Klapuri (2008b) developed a unique "cost-based" approach that began with a definition of what is desirable in a structural description, rather than with a method of extracting the anticipated lines or blocks from an SSM. The space of potential solutions was then searched to maximize the fitness of the description. Their proposed cost function penalized dissimilarity between segments with the same labels, the presence of long unexplained regions, and the use of too many labels.

To maximize such a function over the space of all possible descriptions would be an intractable problem, so familiar methods were used to reduce the search space. Peaks in a novelty function were detected and used as a limited set of possible boundaries. The average dissimilarity between the resulting segments was taken as the "block distance," and the DTW distance between each segment's sequence of vectors was taken as the "stripe distance." Likely segment groupings were detected and combined to form a proposed description. The space of possible solutions was iteratively searched, but by aggressively pruning the search space, a globally optimal solution was obtained in a reasonable amount of time.

#### 2.3.2.5 Applying semantic labels

The analysis at this stage is a segmentation of the audio and a grouping of related or repeated sections, which is conventionally transcribed as a string such as ABCBCCDC. While this is sufficient for many purposes, one may also wish to apply semantic labels: for example, one may want to know which letter in the previous example corresponds to the chorus, which to the verse, and so forth, assuming that these semantic labels are appropriate for a given piece. Most commonly, chorus sections are assumed to be the most frequently repeated segment, or the segment that is repeated most closely; in the case of a tie, the louder group may be chosen, and the other identified as the set of verses. Other means of identifying a chorus are discussed in the "thumbnailing"

discussion under Applications in Section 2.4.1. Methods that seek to apply semantic labels to each section are described here.

Maddage et al. (2004) made use of particularly strong assumptions about structure to generate a description, restricting the output to one of three possibilities. Abbreviating intro, verse, chorus, bridge, and outro as I, V, C, B, and O, respectively, they were IVCVCCO, IVVCVCBCCO, IVVCVBCCO. All sections other than the intro and outro were also assumed to be either 8 or 16 measures long. Shiu (2007) assigned all repeating parts the labels V and C, reserving the labels I, B, and O for non-repeating segments occurring at the beginning, middle, and end, respectively.

Zhu et al. (2005), who applied Goto's RefraiD method separately on SSMs based on melody and lyric features, assumed that the most common section with the same melody and lyrics was the chorus, while the most common section with the same melody but different lyrics was the verse.

Paulus and Klapuri (2010) have proposed arguably the most advanced semantic labelling system, involving supervised learning. They trained a hidden Markov model (described in Section 2.3.3) to learn the most likely sequences of semantic labels, and used this model to generate semantic labels for the estimated analysis. They also experimented with incorporating the semantic labelling step into the cost-based search for the optimal analysis, with slightly improved results.

### *2.3.3 Bottom-up approach: Clustering*

In the bottom-up approach to music structure analysis, feature frames are compared to one another and clustered into groups, sometimes in a hierarchical fashion, eventually building up the final sections and their grouping. Besides straightforward agglomerative and *k*-means clustering techniques, the most important analysis tool in the bottom-up approach is the hidden Markov model, defined below. In this section we describe the various ways these techniques have been used to build structural analyses.

#### 2.3.3.1 Simple clustering

As part of their work on generating audio thumbnails, Logan and Chu (2000) presented two clustering methods for structure analysis. In their first approach, they initially merged the short feature frames into one second-long groups, taking the mean

and covariance of each feature value. Agglomerative clustering was then performed directly on these groups: the groups were exhaustively compared to each other (using the symmetrized KL distance), and if two groups were more similar than a predetermined threshold, they were combined into a single cluster. Note that there was no requirement for these clusters to be contiguous in time. This process repeated until none of the remaining clusters were judged to be similar to each other. The end result was a set of clusters and a mapping from each feature frame to a cluster. Neighbouring frames assigned to different clusters were regarded as segment boundaries, and the cluster assignments as the segment labels.

For their second approach they employed a hidden Markov model. A Markov model is defined by a set of states which a time-evolving process may occupy, and by the probability of switching between any two states, called the transitions probabilities. For instance, in Logan and Chu's approach, the states could be the section labels, and the transition probabilities the likelihood of switching between two sections at any time. In a hidden Markov model (HMM), the states are not directly knowable, but are assumed to probabilistically generate observations at each point in time. The Baum-Welch algorithm is a means of estimating these output probabilities, as well as the likely transition probabilities of the model, and its output may be used to "decode" the hidden state structure from the observation sequence. In Logan and Chu's example, the HMM would estimate a set of probability distribution functions, each associated with a section label and each generating feature vectors with some degree of variation; decoding the HMM would require assigning each feature frame to a state, resulting in a structural analysis. Note that the number of states must be specified as an input to the model.

Aucouturier (2001) used an HMM to cluster feature frames, but this was done explicitly to obtain a mid-level representation of timbre rather than a structural analysis. This is because, while, inasmuch as each section in a piece is defined by its unique instrumentation, the above methods could produce a structural analysis directly, the result is more likely to be highly fragmented. Aucouturier trained an HMM on timbral feature to associate each state with a timbral mixture. In his experiments, this produced some musically meaningful results: in one example using a three-state HMM, the estimated frame labels successfully grouped the frames into three distinct classes: one featuring

vocals with instrumental accompaniment, another featuring the accompaniment alone, and the third containing frames of silence. By labelling each frame according to its estimated state, one obtains a "texture score" which may be searched for patterns; for instance, one could produce an SSM, from which repeated patterns could be extracted according to familiar techniques.

One shortcoming of the clustering methods presented so far is that they do not account for the sequential nature of musical signals. Peeters et al. (2002) formulated an approach that addresses this in two ways: first, they devised "dynamic features," described in Section 2.2.4, which reflect the periodicities with which feature values evolve in time; second, they used a multi-pass clustering method that is able to enforce stronger transition costs in an HMM while providing a good clustering. The first pass segments the piece into regions of relative homogeneity with respect to the dynamic features derived earlier. The average values for each of these regions are used to seed a *k*-means clustering algorithm (described in Section 2.3.1.4), which is adept at merging redundant clusters. The estimated number of clusters as well as their means are in turn used to seed an HMM, which is able to smooth out the description because of the penalty on state transitions.

Kim et al. (2006) took an unusual approach: they conducted an initial segmentation of the piece by applying a tempo-estimation algorithm. Such algorithms are notoriously prone to making octave errors, but Kim et al. treated these errors as informative labels, treating the blocks with different estimated tempi as initial segments. These segments were then iteratively clustered, using a DTW estimate of segment similarity, until a predefined number of segments was obtained.

#### 2.3.3.2 HMM decomposition with many states

Also seeking to address the potential for over-fragmentation when using HMMs, Abdallah et al. (2005 and 2006) presented a sophisticated extension to Aucouturier's work that has formed the basis for many subsequent approaches. They also used an HMM to devise a texture score-like mid-level representation, but theirs contained on the order of 40 or 80 states, rather than 3 or 4 as Aucouturier had suggested. In this way, there is a sufficient variety of states to produce a texture score that can reflect the subtle organization of timbres within a piece.

The HMM states were combined to form state histograms for small neighbourhoods, and these histograms were treated as feature vectors in two subsequent clustering algorithms. One was similar to agglomerative clustering: the KL divergence was calculated between each pair of histograms, with similar pairs iteratively assigned to the same cluster. As a second method, they used a version of *k*-means clustering that interprets the feature vectors as probability distributions over the space of histograms, rather than as feature values.

Several authors have adopted the approach described by Abdallah et al. (2005), using different clustering algorithms, incorporating extra constraints, and so forth. Levy et al. (2006), having obtained HMM state histograms, applied a version of *k*-means clustering that seeks to produce clusters that are coherent, with well-defined reference histograms, and that lead to a temporally contiguous segmentation, where relatively few neighbouring histograms belong to different clusters. While their algorithm also required specifying the number of clusters *M*, it was capable of discarding unnecessary clusters, so that only a sufficiently high value for *M* is required; in one published example, a single piece, whether analyzed using 8, 10, or 12 clusters, consistently led to analyses with only 6 clusters occupied.

One particular focus has been on constraining the length of estimated sections. Rhodes et al. (2006) incorporated a preference for sections with lengths near 20 seconds. Levy and Sandler (2006), on the other hand, observed that they could improve results by assuming that phrases were multiples of 8 measures long. In a post-processing step, they modified the positions of the estimated boundaries to conform to this assumption while minimizing the amount of adjustment required. In a separate publication, Levy and Sandler (2006) used a hidden semi-Markov model (HSMM), which is capable of modelling the state durations implicitly (whereas the HMM transition probabilities model the state durations only implicitly). They also calculated the autocorrelation of the HMM histograms over the entire piece, interpreting the first peak in the graph as the most prominent phrase length; this value was used as an input parameter for the HSMM.

HMM states are usually estimated from timbral features, but Levy et al. (2007) derived a second sequence of states using an HMM-based chord-transcription system. The sequence of timbre and pitch HMM states defined a joint sequence of histograms that

was subsequently clustered. Cheng et al. (2009) developed a variation of the HMM histogram method which made novel use of lyrics data. The stanzas of a song's lyrics were compared to each by finding the longest common subsequence (LCS) of each pair of stanzas; if the LCS of a pair was greater than 60% of the length of each stanza, they were assumed to have the same label. They then applied Levy and Sandler's (2008) method of analysis, but then only retained the segmentation: the labels were discarded in favour of the labels estimated from the lyrics, with simple heuristics to handle cases where the number of segments obtained from each procedure did not match.

#### 2.3.3.3 Other approaches

Although Rhodes and Casey (2007) began by estimating HMM states, they treated this mid-level representation as a string, in a manner similar to Dannenberg and Hu (2002). The HMM state sequence was estimated using 5 states, instead of the usual 40, and the state assigned to each frame was treated as a single character in the string which describes the song. The algorithm searched for the longest approximate pair-wise matches, and then methodically dissected the transitive relationships implied by the detected matches to generate a structural description.

Barrington et al. (2009) applied a technique developed for video analysis as a way to take advantage of both the states and sequences interpretations of musical structure. In their approach, music is seen as a dynamic texture mixture (DTM). A DTM learns from the signal a set of underlying states that produce the feature frames—that is, it estimates the sections in the music that are composed of different musical ideas which comprise the musical surface. However, rather than doing so by clustering the feature frames themselves, the DTM learns to cluster *sequences* of frames. In this way, it learns both the absolute nature of the feature frames (to distinguish different instrumentation mixtures) as well as the repetitive sequential nature of the sequences (to distinguish different rhythmic patterns).

## 2.4 Applications for structural analysis

A structural analysis is abstract, high-level information about a piece that describes or implies much about its content. Because of this, structure analysis has diverse practical applications, which are described in this section. These applications

include automatically producing summaries of audio files, retrieving pieces with similar structure to a given query piece, visualizing structural information, and many more.

### *2.4.1 Thumbnails and summaries*

One of the most important applications of automatic structure analysis is the production of audio thumbnails or summaries, short excerpts that can be auditioned to quickly get a sense of a piece. Online music retailers such as the iTunes and Zune music stores often provide such summaries for their customers. By always choosing summaries that contain, say, part of the chorus, retailers can hope to better promote their products and have more satisfied customers. Ultimately, the factors that make a thumbnail most appealing are subjective, and would have to be discovered with a user study, but as Peeters et al. (2002) point out, with an accurate structural description of a piece, one could ostensibly generate a thumbnail to any specification. For instance, consider a piece with structure ABCBCCD: if choruses make the best previews, the most frequently repeated part of the song may be used (C); on the other hand, the summary could consist of examples of all the transitions between sections (AB, BC, CB, and CD), or of one example of each section (ABCD), and so forth. Considering applications beyond music stores, the latter representation may be useful in a music similarity retrieval task in which songs similar to a query song are requested from a collection. In such a task, audio summaries may be used instead of full songs to speed up the feature extraction and similarity computations (Sandler and Levy 2007).

Identifying the chorus section or the most important section[9] can be done in many ways. Chai (2003) simply chooses the most frequently repeated section as a thumbnail. Levy et al. (2006) identify the most frequently repeated section as the chorus; in the case of a tie, the chorus is the section with higher acoustic energy (i.e., the section that is louder on average). The second instance of the chorus is selected as the thumbnail, based on the observation that the first instance of a chorus may often be an altered version. Goto (2003a), after detecting all repeated sections, identifies as the chorus that section

[9] Although the definition of "chorus" and "thumbnail" clearly differ, in practice, as an application of automatic structural analysis, they are treated as synonymous.

which is of an appropriate length (roughly 8–40 seconds), which occurs preferably near the end of the piece, and which is itself composed of two repeating subsections.

Thumbnails may be extracted without estimating a structural analysis first. Bartsch and Wakefield (2001) extracted a thumbnail by simply selecting the brightest element in their SSM, subject to two constraints: that the element represent a repetition at a lag of at least one tenth the length of the song, and that the element not occur in the last quarter of the song. Van Steelant et al. (2002) independently proposed a similar technique. Cooper and Foote (2002) summed each column of an SSM; the maximum of the result, which indicated the segment that was the most similar to the rest of the song, was selected as the beginning of their thumbnail.

Without estimating a full structural analysis, Wellhausen and Höynck (2003) located the chorus by searching an SSM for three transitively related sections that were at least 5 seconds long. (Recall that when a chorus occurs three times in a piece, three lines will appear in the SSM, reflecting the three pairs of choruses.) Eronen (2007) used a method similar to Goto's (2006a) to extract repetitions, and a method similar to Wellhausen and Höynck to identify the chorus. However, he also considered the proximity of the repeated section to the one-quarter mark of the piece, which he theorized as a likely time for a chorus to occur. Once the chorus was identified, its estimated position was fine-tuned with the assumption that the chorus constitutes two repeating halves; Eronen correlated the relevant portion of the SSM with a kernel that was 32 beats long and had sub-diagonal stripes at a lag of 16 beats.

Logan and Chu (2000) clustered all frames within a piece into groups, and chose the largest cluster as the best group for the summary; the longest contiguous region belonging to that cluster (and occurring in the first half of the piece) was chosen as the thumbnail. Xu et al. (2002) presented a similar approach, and later presented a more elaborate thumbnail selection method which returned multiple sub-summaries representing the different groups created at the clustering stage (Xu et al. 2004, 2005). Each sub-summary was restricted to a particular length, and extended or trimmed depending on which neighbouring frames were more commonly repeated. Lu and Zhang (2003) used a clustering algorithm and searched for the loudest and most frequently repeated sections, deeming these to be the most salient for musical summaries.

Extra steps may be taken to make the extracted thumbnail more pleasant to listen to. For instance, Lu and Zhang (2003) estimated likely boundary locations to ensure that the extracted summary did not cross a structural boundary. To merge a series of short excerpts into a pleasant summary, Peeters (2004) applied a beat-tracking algorithm to match them rhythmically, cross-fading them over 4 beats. Zhang and Samadani (2007) developed a system that would generate thumbnails for songs in a user's collection and present them to the user, who could reject the song or add it to their playlist. Thumbnails were extracted by computing a structural analysis and choosing the first 8 seconds from each section.

### *2.4.2 Retrieval*

Music structure has been used in several cases for music retrieval tasks: for example, retrieving pieces from a collection that are similar to or the same as some query piece. A piece's structure is in some respects a generic feature: many pieces have ABA form, and according to the UPF Beatles data set (see Chapter 4), seventeen songs out of the Beatles 180-song studio catalogue have the structure "intro, verse, verse, bridge, verse, bridge, verse, outro" (Paulus and Klapuri 2010). Nevertheless, the long-term structure of a piece can still be exploited to retrieve pieces of music that are versions of or similar to some target piece in large collections.

To compare two pieces, Foote (2000a) calculated the average acoustic energy over time for each, and compared the two with dynamic programming. The alignment cost between a query piece and each piece in the collection was used to rank the results. Because it seeks an optimal alignment, this method is robust to tempo deviations between pieces. Aucouturier and Sandler (2001) proposed using Aucouturier's (2001) "texture score" as a mid-level representation for retrieval applications. The texture score identifies frames with timbres, and parameterizes each piece as a string; the edit distance between pairs of pieces could thus be used to rank similar pieces.

Gómez et al. (2006) proposed a technique similar to Foote's, but used chroma features that were normalized by the detected key of the piece so that their method could detect cover songs in a different key. They also experimented with only aligning automatically extracted summaries: they performed a structural analysis in the manner of

Ong et al. (2006), chose summary segments for each song according to simple heuristics, and then used the alignment cost between them to rank the similarity of pieces to a query piece. Marolt (2006) used a representation of melody to isolate repeated patterns in a piece; two pieces were then evaluated by comparing all the constituent patterns and recording the two best matches. Potential transpositions were accounted for by finding that transposition that maximized the between-song similarity.

Bello (2009) sought to cluster pieces in a corpus into groups of similar pieces. He generated an SSM for each piece in a corpus, and compared them to one another using the normalized compression distance metric, which measures how much information is shared between them.

Retrieval at a sub-song scale involves locating where in a given piece a short snippet of music, the query, occurs. Izumitani and Kashino (2008) used dynamic programming to discover where the SSM of a query snippet matched the SSM of the target piece. Martin et al. (2009) compared Izumitani and Kashino's algorithm with another image-alignment algorithm, and adapted them to process symbolic queries such as "ABAC," with and without information about the relative duration of these sections.

### *2.4.3 Visualization and navigation*

Visualizing a piece's structure may be helpful to a music theorist conducting an analysis; Foote's (1999) original article on SSMs recommended them as a visualization tool for just this purpose. Wolkowicz et al. (2009) presented a system called Midivis, which uses several colour-coded SSMs to visualize the similarity between different parts of a piece encoded in MIDI format. Once musical structure has been estimated from an SSM (or other representations), the piece may be visualized in a more concise way. Goto (2003b) created a system called SmartMusicKIOSK, based on his RefraiD method, that shows separate, repeated sections in a piano roll-style notation (see Figure 2.10). Zhang and Samadani (2007) demonstrated an interface that would show the computed structure of a piece and present the user with a thumbnail for each section. A plug-in to the Sonic Visualiser analysis environment will analyze the structure of the audio and display the result as colour-coded regions (Sandler and Levy 2007).

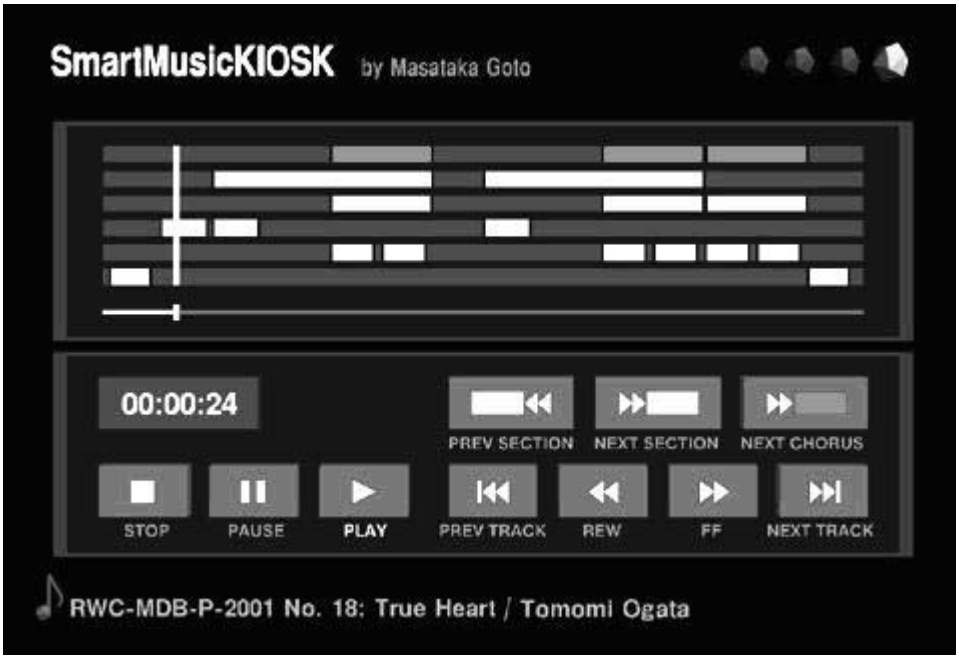



Figure 2.10. SmartMusicKIOSK screenshot. Piano roll-style notation shows in each row all repetitions of a single event. <http://staff.aist.go.jp/m.goto/SmartMusicKIOSK/index.html>, accessed 10 August 2010.

Structural information can also be used to streamline a user's audition of a sound file. For instance, if a media player had knowledge of an audio file's important large-scale structure, the fast-forward function could be replaced by a function that skipped to the next section, allowing a swift perusal of the track. SmartMusicKIOSK includes this functionality, as well as SyncPlayer, a multi-modal music navigation software developed by Fremerey (2006). Fazekas and Sandler (2007) developed an add-on to the Audacity audio editing environment that uses a version of Levy and Sandler's structure analysis system to analyze structure in a multi-track recording environment, estimating important breakpoints. In this case, structural analysis facilitates the user's own editing of a sound file. Eronen (2009) presented a prototype application that allows similar advanced navigation, as well as a visualization of an SSM which may assist the user in editing an automatically extracted thumbnail to be used as a ring-tone.

### *2.4.4 Other applications*

#### 2.4.4.1 Alignment

Structure analysis can be used to constrain or improve the alignment between different files. For instance, to time-align two different versions of the same piece, Müller and Appelt (2008) generated an SSM from the concatenation of each piece's feature vectors. The structure of the pieces was then jointly determined using the same structural analysis method as Müller and Ewert (2008). Since the two versions were potentially quite different from each other, the method they had developed for enhancing and detecting line segments in the SSM in the face of tempo variations was especially useful for this task. The estimated structure analysis was then used to constrain the alignment between different versions.

In a similar vein, Lee and Cremer (2008) proposed a system for aligning lyrics with an audio file. They applied the structure analysis method put forth by Cooper and Foote (2003) to divide the audio into labelled segments, and used dynamic programming to choose the best alignment between these segments and the paragraphs in the lyrics.

#### 2.4.4.2 Compression

Signal compression can be achieved by encoding two regions of an audio file with the same signal. To do so, Cunningham and Grout (2009) searched a piece exhaustively for similar regions that could be identically encoded. They described using an SSM and beat information to improve the efficiency of their compression algorithm by greatly reducing the search space for potentially matching signals. Jehan (2004), after segmenting a piece with a note onset detection method and generating an SSM, grouped the resulting segments using *k*-means clustering. Each group was represented only once in the compressed audio file, and the parameter *k* was chosen depending on the desired accuracy and compactness: lower values of *k* would lead to smaller file sizes at the expense of fidelity.

#### 2.4.4.3 Boot-strapping other MIR tasks

Chord labelling, beat-tracking, and tempo estimation have been incorporated into structure analysis techniques. Conversely, structure analysis has been used to improve the

robustness of these other techniques. For beat tracking, Dannenberg (2005) detected similar segments with an SSM and used these to constrain an algorithmic search for a globally optimal solution to the beat-tracking problem that gives structurally similar regions similar beat structures. Foote and Uchihashi (2001) developed a method for estimating tempo from an SSM by producing a beat spectrum: a function was created by summing along the diagonals of the SSM, and the autocorrelation of this function was used to provide a tempo estimate. In that study, they also demonstrated the use of a novelty function to detect note onsets. Mauch et al. (2009) obtained a structural analysis and then, considering each instance of a particular part (e.g., all the verses), averaged together the features used to estimate a chord analysis. The chord labels estimated from the average were used to label each part.

#### 2.4.4.4 Video applications

Foote and Cooper (2003) have demonstrated the use of SSM techniques for automatic video analysis. In the same manner that Foote (2000b) analyzed music, Cooper and Foote (2001) implemented a scene boundary-detection algorithm in which a novelty function was computed from an SSM created using video frame intensity histograms as a feature. Later, Foote et al. (2002) developed a system that detected well-shot portions of home videos and synchronized them to audio, automatically aligning scene changes with musical boundaries.

#### 2.4.4.5 Speech/music and vocal/instrumental discrimination

Another field related to structure analysis is speech/music or vocal/instrumental discrimination. For example, one may wish to automatically divide a field recording into sections and provide labels such as “speech,” “solo singing,” “instrumental,” and so forth (e.g., Marolt 2009), or divide a radio broadcast into music and non-music sections (e.g., Goodwin and Laroche 2004). We will not consider such research in this thesis, but vocal/instrumental section discrimination is sometimes considered relevant to structure analysis, since verse and chorus sections are usually sung while intro, bridge, and outro sections may be instrumental. As noted already, it has been used to help detect boundaries (e.g., Su et al. 2009) and produce full structural analyses (e.g., Maddage et al. 2004; Xu et al. 2004).

#### 2.4.4.6 Watermarking

Xu et al. (2007) made use of structural information to improve an audio watermarking algorithm. Like a watermark in a piece of paper, an audio watermark is supposed to embed information into an audio stream that is robust to simple audio manipulations (so that it cannot be easily removed) but will not disrupt a listener's experience. Their system performs a note-level segmentation and classifies each segment as being a vocal or instrumental section, embedding bits of the watermark differently for each type.

#### 2.4.4.7 Pattern recognition

Ong and Streich (2008) presented a technique for extracting short repeating "loops" from recordings. The technique is similar to using a checkerboard kernel to locate boundaries: after generating a time-lag SSM and estimating the probable loop period, they correlate the SSM with a two-dimensional filter that will match regions based on a loop with that period.

Although this application is directed at composers and remixers who may wish to efficiently extract reusable snippets from recordings, it relates to the field of pattern recognition, which is more generally useful for musicological analysis. For example, see the analysis of the song "C'était bien" in Aucouturier (2001), and of "Naima" in Dannenberg (2002). Recognizing large-scale structure is important for algorithmic composition, a point made by Eck and Schmidhuber (2002) and Jehan (2005a).

## 2.5 Summary

This chapter began with a brief discussion of the state and sequence approaches to structural analysis. The literature on this topic was then surveyed in two parts: the first dealt with the broad category of top-down approaches, which usually involve computing a self-similarity matrix in order to locate repeating states of sequences. The various exploitable properties of these matrices were discussed, as well as the myriad techniques used to process it. The second part dealt with bottom-up approaches in which clustering techniques are used to build a model of the representative states of the piece. Finally, practical applications for structural analysis were discussed, including audio thumbnailing, music retrieval, and intelligent audio navigation.

# 3 Algorithms

Five algorithms were evaluated in this experiment. Three of these, Peiszer (2007), Levy and Sandler (2008) and Barrington et al. (2009), produce full structural analyses, segmenting the piece into large-scale sections and providing group labels. The other two, Mestres (2007) and the Echo Nest API, perform only the segmentation step, producing a list of boundaries between the sections of a piece. In this chapter, the analysis techniques used by each algorithm are described in greater detail than in the previous chapter, although these descriptions assume a familiarity with the feature extraction methods described fully in Section 2.2 and the basic analysis techniques described in Section 2.3. An account of how the algorithm was implemented in this thesis, including the parameter settings used, appears at the end of each section.

## 3.1 Peiszer

The system created by Ewald Peiszer, described in detail in his thesis (Peiszer 2007) and summarized by Peiszer et al. (2008), combines the top-down and bottom-up approaches. Like Cooper and Foote (2003), it operates in two steps: first, candidate segment boundaries are estimated using a novelty function derived from a SSM. Second, these segments are grouped according to information obtained by a clustering algorithm that operates on low-level frames.

### *3.1.1 Features*

Peiszer experimented with several feature sets, including MFCCs, chroma vectors, rhythmic features, and raw spectrogram coefficients. In each case, the full, raw output was used: all 40 MFCCs were kept; the unwrapped chromagram vector was used (i.e., contributions from each octave were not binned together); and the raw output of the Fourier transform was used for the spectrogram coefficients. The rhythmic features included Rauber et al.'s (2002) fluctuation patterns (which consist of a matrix of Fourier coefficients of each of 24 critical bands), Peiszer used a set 7 spectral descriptors for each of these bands as another feature set. Each feature was calculated over beat-aligned windows using the beat locations estimated by Simon Dixon's (2001) BeatRoot software.

### *3.1.2 Segmentation*

Each feature set was used to generate a SSM using the Euclidean distance function, from which a novelty function was calculated using the standard Gaussian-tapered checkerboard kernel as described in Section 2.3.2.1. Again, Peiszer experimented with his analysis parameters: he varied the size of the checkerboard kernel and applied various filters to smooth the novelty function. He also applied a variety of heuristics to pick peaks from the result, including a minimum threshold, a minimum inter-peak distance, and a requirement that boundaries be located at the downbeats of measures. He found that segmentation performance did not vary significantly with the analysis parameters, and that the essayed heuristics did not improve overall performance.

### *3.1.3 Structure analysis*

Having obtained a set of boundaries, the second part of the algorithm sought to label these sections with some clustering method. Several clustering methods were used, including means-of-frames clustering, hierarchical agglomerative clustering, a voting method, and a dynamic time warping (DTW)-based method.

In the means-of-frames approach, each segment was represented by the means of its feature values, and these values were then grouped using *k*-means clustering. All of the segment means that were nearest to the same centroid were given the same label. In the agglomerative clustering method, the segments' mean values were iteratively grouped together, beginning with the most similar segments, until the desired number of clusters had been obtained; complete linkage was used, meaning that the maximum dissimilarity between any two component frames from two segments was taken as the dissimilarity between these segments. The voting approach began with the means-of-frames procedure described above to define the cluster centroids, except that segment labels were assigned by estimating the group to which each segment's constituent frame belonged, and having them vote on a final label. Finally, the DTW approach recalled Cooper and Foote's (2003) approach: a segment-indexed SSM was generated using the DTW alignment cost of each pair of segments; *k*-means clustering was then used to assign segment labels.

Each of these methods required the presumed number of segment types as an input parameter. Attempts to automatically estimate the correct number of distinct labels

using cluster validity indices, as well as experiments involving user input for this step (simulated by choosing the number of labels in the song's ground truth), did not significantly improve results. The different clustering methods all had similar success, except that the voting method performed significantly worse.

### *3.1.4 As used in this thesis*

The code for Peiszer's analysis system, implemented in MATLAB, was obtained from Peiszer's website[10]. The updated version of BeatRoot (Dixon 2007), also freely available online, was used to obtain the required beat locations. Peiszer's algorithm was executed using both MFCC and chroma features, and using all clustering methods except for voting, which had performed the least well. Values for the expected number of clusters *k* ranged from 3 to 10. The segmentation step and the clustering algorithms were always executed with the same parameters: the novelty functions were generated from a 96-pixel wide Gaussian checkerboard kernel and smoothed using an 8-sample long moving average filter, the SSM was calculated using the Euclidean distance metric. These parameters had led to good results in Peiszer's evaluation. I elected not to repeat Peiszer's full exploration of the large parameter space, since his experiments suggested that little advantage could be had from tuning the parameters.

## 3.2 Levy and Sandler

Levy and Sandler's (2008) algorithm follows the general state-based approach proposed by Abdallah et al. (2005) and developed by others at Queen Mary, University of London. A hidden Markov model (HMM) is used to recast the audio as a sequence of labelled timbre-types, and local histograms of these types are clustered to achieve a structural analysis.

### *3.2.1 Features*

Levy and Sandler used the MPEG-7-defined AudioSpectrumEnvelope descriptor as a timbral feature, from which they derived AudioSpectrumProjection feature vectors. For each audio frame, the envelope was represented by the power level in each of several

[10] <http://www.ifs.tuwien.ac.at/mir/audiosegmentation.html> accessed 20 February 2010.

logarithmically-spaced frequency bands. They used a one-eighth octave spacing for frequencies between 62.5 Hz and 16 kHz (along with two more bands for frequencies outside this range), giving a 66-dimensional feature vector. Principal component analysis was used over the full piece to reduce this to 20 dimensions, giving the AudioSpectrumProjection vector. The spectral envelope of each frame was normalized by its overall power, which was preserved as a 21st dimension. The relatively high frequency resolution in this procedure preserved both timbre and pitch information, so this feature is referred to by the software as a "hybrid" CQT feature. Levy and Sandler also experimented with MFCCs and with 12-dimensional chroma vectors, although they found that the chroma led to poorer results.

### *3.2.2 Structure analysis*

Their algorithm proceeds in two steps. The first achieves a low-level labelling, and the second applies a clustering algorithm to estimate a set of $k$ section types. To produce the low-level labels (or, seen another way, a high-level feature extraction), a 40-state HMM is applied to the beat-aligned feature frames to estimate a set of 40 "timbre types." The piece is then decoded with the Viterbi algorithm so that each frame is labelled with one of these types. Finally, local histograms of state labels are obtained by tallying how often each state label occurs within neighbourhoods of 7 frames (i.e., within 7 beat lengths). Although different sections of a piece almost certainly contain some frames with the same label, the relative frequency of each state label, revealed by the histograms, can be characteristic of each section.

These histograms are treated as high-level feature vectors in the clustering analysis that follows. Following the notation in Levy and Sandler (2008), the goal is to associate each histogram $x_n$ in the sequence of N histograms with its section-type label $y_n$. The cluster centroids $m_k$ must also be estimated as part of the analysis, and the number of centroids $k$ is a required input parameter. Levy and Sandler introduced a constraint to favour contiguous sections of a minimum length: a function measuring the likelihood of a particular sequence of histogram labels included a penalty term for whenever two nearby frames within some neighbourhood $d_{ML}$ were given differing labels.

This problem may be solved with an expectation-maximization (EM) algorithm. EM is an approach that iteratively improves upon an estimate for the unknown model parameters $m_k$ (collectively indicated by $\Theta$) and the hidden state labels $y_n$ by alternately applying two steps: in the expectation step, the current best-guess set of centroid parameters $\Theta$ are used to predict the likelihood of the sequence $y_n$; in the maximization step, these values for $y_n$ are used to revise the estimate of $\Theta$. Since EM is sensitive to the initial estimates of the unknown parameters, Levy and Sandler initialized $\Theta$ with the output of a conventional $k$-means clustering algorithm. Because the additional constraint on the segment length makes the execution of the standard EM method impractical, Levy and Sandler modified the algorithm to obtain an approximate solution. The output of the modified EM algorithm is a label for each frame, and the structural analysis is completed by taking contiguous sequences with the same label as the piece's labelled sections.

### *3.2.3 As used in this thesis*

Mark Levy implemented the algorithm described above as a Vamp plugin[11] that may be accessed by Sonic Annotator[12] (Cannam et al. 2010). Created at Queen Mary, University of London, Sonic Annotator is a publicly available command-line tool that can extract low-level features from audio files, such as chroma vectors and MFCCs, and automatically annotate files with beat locations, polyphonic transcriptions, and other information.

The version of Levy and Sandler (2008) implemented as a plugin can accept three feature sets: MFCCs, chroma, and CQT. They remarked that informal testing showed that chroma were less effective features than the others, so only the CQT and MFCC options were used in this evaluation.

Two other parameters accepted by the plugin are the number of clusters, $k$, which was set to values between 3 and 10, and the minimum segment length $d_{ML}$. Whereas this parameter was set as either 8 or 16 beats by Levy and Sandler (2008), it must be specified

---

[11] <http://www.vamp-plugins.org/plugin-doc/qm-vamp-plugins.html#qm-segmenter> accessed 27 July 2010.
[12] <http://omras2.org/SonicAnnotator> accessed 27 July 2010.

as a length in seconds in the plugin, since the beat-alignment step was not implemented. For this evaluation, values of 4 and 10 seconds were used.

## 3.3 Barrington et al.

Aucouturier (2001) noted that a central weakness in his HMM-based clustering approach to music structure analysis (similar to the first step of Levy and Sandler's algorithm, described above) was that it did not account for the sequential nature of music, so that important melodic motifs or bold changes in rhythm may be ignored by this approach. This, in a nutshell, is the gap between the states and sequences views of music structure. Barrington et al. (2009) attempted to close this gap, applying a technique developed for video analysis called dynamic texture mixture (DTM) modelling.

### *3.3.1 Dynamic texture mixture model*

Described briefly in Section 2.3.3, a DTM is a way to model a piece of music as a series of regular but time-evolving sound textures. While this approach retains the ability to distinguish sections of a song based on which timbres are present, it is designed to also distinguish two sections composed from the same palette of timbres but arranged in a different order. In this way, one may identify a section based on its distinct drum pattern, even if the individual drum sounds occur elsewhere in the song; similarly, with a different feature input, this method may distinguish two sections that have the same chords but differing harmonic rhythms. Barrington et al. used both MFCCs and chroma vectors as features.

Following Barrington et al.'s (2009) notation, we may define a DTM as a model for generating feature vectors based on the following system of equations:

$$x_{t+1} = A_z \cdot x_t + v_t$$

$$y_t = C_z \cdot x_t + w_t$$

The observed variable $y_t$ is given by the feature vector at time $t$, while the hidden-state variable $x_t$ encodes higher-level information about the sound's timbre and how it changes in time. The evolution of $x_t$ is controlled by the state transition matrix $A_z$, which is different for each dynamic texture $z_t$, and the observation matrix $C_z$ translates these into

the observations $y_t$. The two functions $w_t$ and $v_t$ represent the Gaussian noise present in these processes. Whereas in an HMM each frame is assigned to a state $i$, the assignment of each frame to a dynamic texture is given as a vector $z$ where each element indicates the probability that the segment originates from a particular dynamic texture.

The goal is to estimate a sequence of hidden-state variables $x_t$, to learn a set of $k$ dynamic textures that control the evolution of $x_t$, and to estimate the dynamic texture being sampled at each instant. The dynamic textures are each defined by a parameter set $\Theta_z$ that includes $A_z$ and $C_z$, as well as other parameters defining the texture's initial state and noise properties. All of this information must be learned from the observed variables $y_t$.

This problem may be solved with a version of the EM algorithm described previously (see Section 3.2.2). In this case, the hidden variables are $x_t$ and $z_t$, which are revised iteratively to obtain an estimate of the texture parameter sets $\Theta$. Note that the method is sensitive to the initial estimate of $\Theta$, and is only guaranteed to find a local maximum.

### *3.3.2 Structure analysis*

The DTM model approach can accomplish a full analysis in a single step: Barrington et al. divided the song into long, 5-second audio sequences with 90% overlap, and input these sequences into a DTM model to estimate a series of dynamic textures using EM. The labelling was provided by choosing the most likely label $z_i$ at each time $t$, which, as with Levy and Sandler, implicitly determines a segmentation.

Such coarse windowing is necessary to model long-scale musical patterns, but can lead to poor boundary estimation. Barrington et al. therefore implemented a fine-grain analysis to refine the boundary positions. The previously estimated DTM was used to assign shorter sequences (1.75 seconds long with 97% overlap) to dynamic textures, and the boundaries found in the coarse analysis were updated by simply shifting them to the nearest boundary in the fine-grain analysis.

Barrington et al. (2010) extended their previous work by introducing constraints on the length of the estimated segments. Like Levy and Sandler (2008), they imposed a

preference for segments of roughly 16–20 seconds, or roughly 4–8 measures. However, this step was not implemented in the version evaluated in this thesis.

### *3.3.3 As used in this thesis*

The MATLAB code for this structure analysis system, along with the DTM modelling algorithm developed by Chan and Vasconcelos (2008) upon which it is founded, was generously loaned to me by Luke Barrington. The algorithm was executed using both MFCCs and chroma vectors as features. However, since the system took significantly longer than the others to process, values for the expected number of clusters *k* were only varied between 3 and 6. Both the coarse and the fine-grain segmentation output were kept for evaluation.

## 3.4 The Echo Nest

The Echo Nest is a company that analyzes music and amasses music-related information from online sources. Some of this information is sold to music marketers, but much of it is provided freely via a developer application programming interface (API). The developer API includes methods for uploading audio files and downloading sophisticated analyses, including the estimated location of beat and measure onsets and the estimated key and tempo of the piece. The get_sections method, which is accessed through the analyze method, estimates the boundaries (and not the labels) of those sections which are "the largest chunks in the track and usually correspond to structural units (verse, bridge, solo)." [13]

Although the audio analysis algorithms used by the Echo Nest are not publicly documented, a post on the Echo Nest developer forum from Tristan Jehan, one of the founders of the Echo Nest, reveals that the get_sections method is based partly on the results of the get_segments method.[14] The latter method divides the piece into short segments (each typically less than a second) described by pitch and timbre features.

One might speculate that these short segments are obtained in the same manner as the low-level segmentation described in Jehan (2005b). In that article, Jehan described

---

[13] <http://developer.echonest.com/docs/v3/track.html#get-sections> accessed 27 July 2010.
[14] <http://developer.echonest.com/forums/thread/70> accessed 27 July 2010.

the use of dynamic programming to construct several hierarchically-related SSMs: peaks in an onset detection function were used to create a segment-indexed SSM; this pattern of onsets provided the basis for estimating a beat-indexed SSM; and this in turn was analyzed to create a pattern-indexed SSM (i.e., where each pixel represents the similarity between two short 2–16 beat sequences). Although Jehan did not describe how to extend this procedure and obtain a section-indexed SSM, he noted that the significant and sophisticated reduction in the size of the SSM that he achieved could lead to improved performance in a structural segmentation algorithm such as Cooper and Foote (2003) or Chai and Vercoe (2003). However, the ultimate method used by the Echo Nest to produce its structural segmentation remains private to the company.

### *3.4.1 As used in this thesis*

Despite only producing segmentations and being a figurative black box, the analyses provided by the Echo Nest remain a useful point of comparison for this study. Using their developer API, each audio file was uploaded to the site as an MP3, and the segmentation returned using the get_sections portion of the analyze method. While the other algorithms used all used higher-quality WAV files, the slight reduction in quality to MP3 was necessary to keep the file size manageable for uploading to the Echo Nest.

## 3.5 Mestres

Xavier Mestres (2007) described in his thesis, on the topic of singer identification, a segmentation algorithm that the author noted was surprisingly effective at reproducing a piece of music's structural boundaries. The algorithm made use of the Bayesian Information Criterion (BIC), which is a way to rate the quality of a model by weighing the likelihood of the data given the model, against the number of parameters in the model and the number of data points used to estimate it. In his case, every model being compared was a Gaussian distribution, the classic bell-curve distribution model which is defined by its mean (i.e., the average feature values of the vectors included in the distribution) and its variance (a measure of how spread out these feature vectors are).

### 3.5.1 Segmentation

The BIC can be used to decide if there should be a boundary at some point $n$ in a short sequence of $N$ feature vectors by considering two models. The first model supposes that the entire sequence of feature vectors originates from a single Gaussian distribution, whereas the second supposes that the two parts (divided by the point $n$) come from different Gaussian distributions. Mestres uses the following equation to determine whether the point $n$ is likely to be a boundary:

$$BIC(n) = N \log|\Sigma| - N_1 \log|\Sigma_1| - N_2 \log|\Sigma_2| - \lambda P$$

where $N_1$ and $N_2$ refer to the number of frames in the first and second part of the sequence of $N$ frames, and each $|\Sigma|$ term refers to the variance of the Gaussian distribution used to model each part. The final term penalizes the BIC score according to the number of extra dimensions in the two-Gaussian model. If BIC($n$) is greater than 0, then the two-Gaussian model has succeeded at reducing the variance even at the cost of the extra dimensions, and it is estimated that a boundary exists at time $n$.

### 3.5.2 As used in this thesis

Mestres' segmentation algorithm operates in three passes: the first achieves a coarse segmentation, the second fine-tunes the placement of the detected boundaries, and the third eliminates unnecessary boundaries for a better global solution.

In the first pass, each sequence of $N$ feature vectors is taken in turn, and the BIC test used to determine whether the vectors from 1 to $n$ and those from $n$+1 to $N$ are better explained by one or two Gaussian distributions. Several values of $n$ are tried for each sequence. In the second pass, the region around each detected boundary in the first pass is retested using values of $n$ that are closer together. The previous steps use relatively small windows, so the resulting segmentation may contain boundaries that are too close together, or spurious boundaries between segments that are actually very similar on the whole. The third pass attempts to eliminate these by applying the BIC test to every pair of adjacent segments and only keeping those which pass it.

The various hop sizes and window sizes were the same as those used in Mestres (2007, page 33): the window size $N$ in the first pass was 10 seconds, with a hop size of 3 seconds; the window and hop sizes were 6 and 0.5 seconds, respectively, in the second

pass; and the minimum segment size was 10 seconds. I did not use the system directly; rather, the execution of this algorithm was kindly carried out by Nicolas Wack at Universitat Pompeu Fabra, and the results were sent to me for evaluation.

# 4 Annotation and Evaluation

As stated in Chapter 1, the goal of automatic music structure analysis is to decompose a piece into its large-scale sections, and to provide labels indicating which sections are similar to or repeated instances of each other. The algorithms described in the previous two chapters all attempt to produce such an analysis (or some component of it, be it a segmentation, or an extraction of only the chorus sections). The performance of these algorithms should be quantitatively evaluated by comparing the analyses they produce to human-created reference annotations. The production of these "ground truth" reference annotations, as well as the evaluation of automatically-generated descriptions, are the topic of this chapter.

In this section we first discuss important problems related to the production of ground truth structural descriptions, and recent efforts that some have undertaken to counteract these problems. Following this is a brief description of the musical corpora and accompanying annotations that were used in the present study. Finally, the various evaluation metrics used to compare the estimated structure descriptions to the annotations are described.

## 4.1 Producing annotations

The algorithms described in Chapter 2 all seek to produce concrete, unambiguous structural descriptions, and in order to evaluate their validity one ought to compare these to the true descriptions. However, music structure is a musicological abstraction that may only be subjectively perceived, and so the "true" structure of a piece, if it exists, can never be unequivocally determined. This is a problem for those who would produce ground truth descriptions, such as those illustrated in Figure 4.1, which are single, absolute descriptions of a piece. Two especially important reasons for this problem are: first, that music theory itself lacks an adequate means to describe musical similarity; and second, that music typically features hierarchical musical relationships.

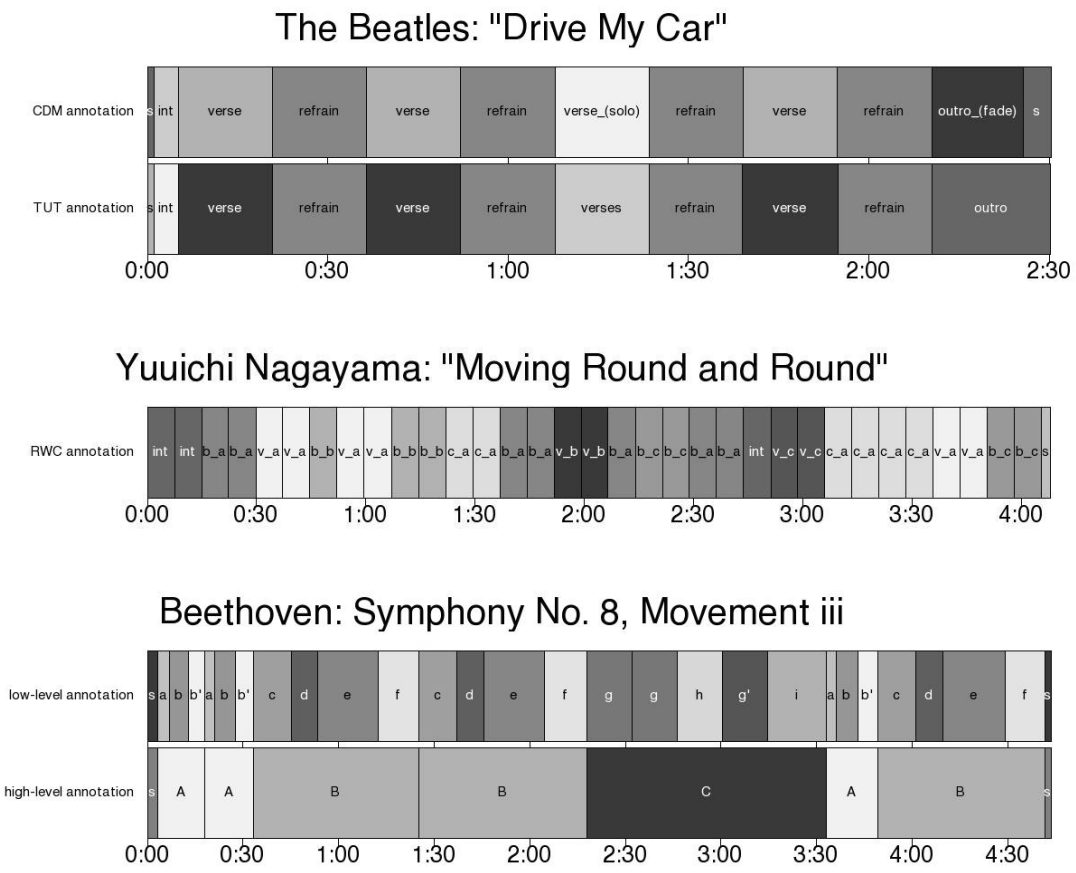



Figure 4.1. Example ground truth annotations for three songs. From top to bottom, the pieces used and the sources of each annotation are: “Drive My Car,” by The Beatles; “Moving Round and Round,” by Yuuichi Nagayama; and the third movement of Beethoven’s 8th Symphony. Two versions of the ground truth are presented for the first and third songs; see Section 4.2 for a description of these data sets. For the Beatles annotations, “int.” stands for “introduction,” and in the RWC annotation, “v_a” stands for “verse_a,” “b_b” for “bridge_b,” and so forth. The labels in the Beethoven annotation are not semantically meaningful.

A structural description, regardless of how it was obtained, constitutes a segmentation of the piece into different sections, and a set of labels to indicate which sections are similar to each other. Theories of music may be called upon to assess the similarity of two passages of music, perhaps comparing them on the basis of their melodic, harmonic, rhythmic, or dynamic content. However, judgements of musical “similarity” are not quantitative, and there is no method to categorically ascertain whether one passage is a repeat of another, or merely similar, or dissimilar. Consider, for instance, a section of a song in which an instrumental soloist replaces the singer for a

verse: should the section use the same label as the verse, due to their harmonic-sequential similarity, or does it merit a new label due to its different instrumentation, and, perhaps, melody? While either answer may seem reductive, one must be chosen in order to quantitatively evaluate a particular algorithm's output. Because of the complexity of musical relationships, ambiguous situations like this arise very frequently.

Another problem with obtaining a single, absolute description of a piece's structure is that it cannot account for the rich hierarchical relationships that music expresses. Long sections of music are usually composed from shorter musical ideas, which can be repeated or varied, and which in turn may be derived from a limited palette of shorter motives. Consequently, a piece of music may be pictured as having a tree-like structure; a potential analysis for the song "I Saw Her Standing There," by The Beatles, is pictured in this fashion in Figure 4.2a.

To choose a single description of a piece's structure amounts to taking a horizontal cut through this hierarchical tree structure. However, at what level this cut should be made is a choice that is usually made intuitively by the analyst, and different analysts may disagree: any single cut will omit the more detailed structure contained in the branches below, and ignore the more abstract view presented in the branches above. Figure 4.2b shows an example of such a disagreement. The annotation offered by the Center for Digital Music (CDM) segments the piece at a finer scale than that offered by Tampere University of Technology (TUT), and it is justify a claim that one is better than the other. (See Section 4.2 for information on these data sets.) Recognizing this, Peiszer (2007) created annotations where some sections included optional subdivisions, creating multi-dimensional structure annotations. In theory, an annotator could extend this process and create a full hierarchical tree for each annotation. However, the hierarchical connections themselves can be subjective. To give one example, two listeners may agree on the same fine-scale structural description (in Figure 4.3, the description "aabbc"), but disagree on how these smaller sections ought to be grouped together (one treats "c" as balancing the two "b" sections at the middle hierarchical level; the other treats "c" as merging with the "b" sections).

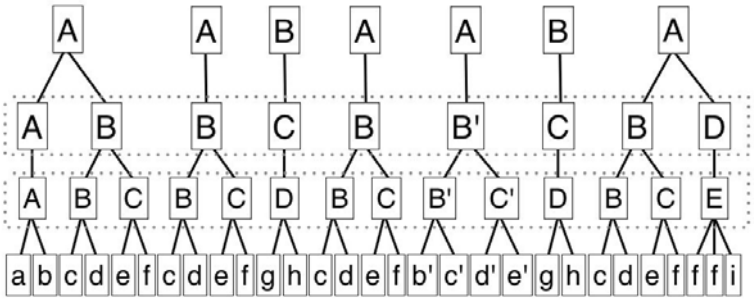



Figure 4.2a. Tree diagram describing the hierarchical structure of the song "I Saw Her Standing There," by The Beatles. Boxes in the same row with the same letter indicate sections of the song with the same or similar music; lines indicate which sections are grouped together at some level in the hierarchy.Boxes with the same letter in different rows are not necessarily related. The upper and lower gray dotted outlines correspond to different analysis of the song provided by two different ground truth sources, CDM and TUT, respectively. See Section 4.2 for information on these data sets.

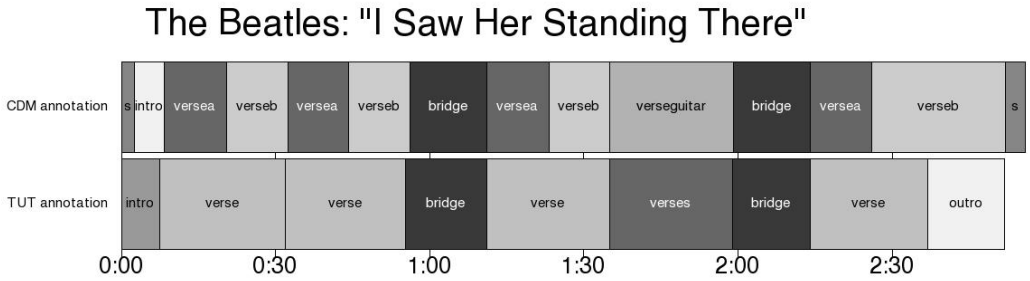



Figure 4.2b. Ground truth annotations provided by CDM and TUT for the same song, "I Saw Her Standing There." Each corresponds to a dotted line in Figure 4.2a

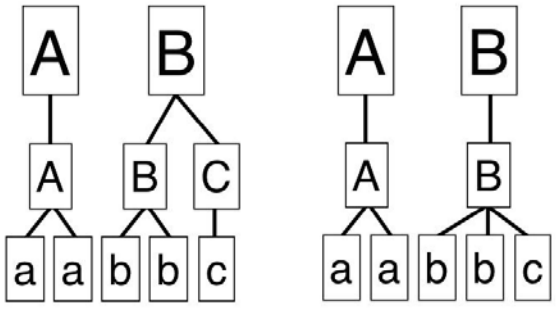



Figure 4.3. Two tree diagrams demonstrating a possible disagreement in hierarchical relationships, despite agreement at one scale. The two versions both describe the same low-level and high-level structure, but disagree on how the small sections are grouped

One way to circumvent these issues is to introduce semantic labels that have musical meaning, such as "verse" and "chorus." The properties of these sections may help establish criteria for making similarity judgements. Furthermore, by defining meaningful sections of music, we may define a specific hierarchical level that is meaningful to us. Five widely-recognized labels are briefly defined below:

- verse: a section that repeats with the same music but different lyrics
- chorus: a section that repeats with the same music and lyrics, often regarded as being more intense or more energetic than the verse
- intro: a non-repeating introductory section, usually without vocals
- outro: a non-repeating concluding section
- bridge: a non-repeating section of contrasting material that occurs in the middle of a song

However, semantic labels introduce problems of their own. As Peeters and Deruty (2009) point out, the fact that semantic labels conflate the notion of musical similarity (embodied in ABAC-style notation) with musical function may result in curious annotation decisions: for example, two sections that sound exactly the same may be given the differing labels "intro" and "outro" depending on their position in the piece. Also, the above list is hardly comprehensive enough to describe all the types of sections that one encounters in popular music; and yet, as one includes more vocabulary terms—"instrumental," "solo," "pre-chorus," "coda," and so forth—the distinctions between them blur, and the absence of precise definitions grows more regrettable. Relying on a vocabulary also requires one to devise a new vocabulary for new genres of music: the above terms would not clarify one's analysis of, say, Bach's Prelude no. 1 in C major.

Another way to overcome the subjectivity of structural analysis could be to collect analyses from a number of listeners and derive from their responses a most agreed-upon structural description. Bruderer et al. (2006) pointed in this direction in their study of boundary perception, in which they had listeners annotate a song by indicating where they felt there were phrase or section boundaries. Their results suggested that while there may be broad disagreement about where most boundaries lie, there was also significant agreement on a small number of boundaries. When asked in a second experiment to rate the salience of candidate boundaries, these agreed-upon boundaries also scored the

highest. Although this research was limited to boundary estimation, the promising results could inspire research to extend this work towards segment labelling.

Perceptual studies aside, the problems described above—the fuzziness of our perception of musical similarity, the presence of hierarchical musical relationships, and the ambiguity of our vocabulary—seem to be fundamentally irresolvable. The sliding perceptual scale between exact repetitions, approximate repetitions, variations, and simply dissimilar sections of music may always resist quantitative description; different listeners may always attune to different levels of a piece's hierarchical structure; and our vocabulary, no matter how refined, may always fail to account for all musical situations. Perhaps because of this, not all regard the subjective nature of ground truth annotations to be an important problem, and in practice, ground truth seems simply to be taken as the judgement of one or two annotators who have listened to a song carefully. Rather than attempt to develop more trustworthy annotations, the response is to simply acknowledge the subjectivity of the ground truth, and to treat the subsequent empirical evaluation with a small amount of skepticism.

However, some very recent research efforts have bucked this trend by seeking to devise structure analysis procedures that are repeatable; that is, where different annotators who apply the same method will reliably generate the same annotations. The boundary-estimation procedure proposed by Bimbot et al. (2010) specifies which layers of musical information are to be considered (e.g., harmony, tempo, timbre, and lyrics), establishes specific criteria for making similarity judgements (e.g., "Could these two segments be swapped while retaining the larger section's musicality?"), and heavily constrains the placement of segment boundaries by setting a preferred segment length and allowing deviations only in specific situations. Their research is ongoing, and again limited to segmentation, but they reported high agreement between annotators in a small evaluation.

Peeters and Deruty (2009), meanwhile, have developed a method for full structural analysis that seeks to clarify the annotation process by separating different layers of musical information: labels referring to function (such as "intro" or "chorus"), to instrumentation, and to the musical ideas themselves are kept in separate layers to avoid conflating these notions. This way, there is no conflict for the annotator when an instrumental solo uses the same music as the verse, or when the music from the

introduction is repeated in the bridge, or when two choruses feature vastly different instrumentation. Again, research is ongoing, but they have used this method to annotate 300 songs already, with high inter-annotator agreement, and a variation of their method has been adopted by the SALAMI (Structural Analysis of Large Amounts of Music Information) project at McGill University, which aims to annotate over 1,000 pieces. Like Peiszer, the SALAMI annotations also include descriptions at two hierarchical levels.

To summarize: there are significant problems with annotating structure, stemming from the fuzzy nature of listener's perception of similarity and from the hierarchical nature of musical structure. Having a vocabulary of semantic section labels can guide a listener towards a narrower space of descriptions, but annotating structure remains a subjective process. However, recent research on novel annotation methods may lead to significantly more robust annotation practices and, hopefully, more meaningful evaluation.

## 4.2 Description of data sets

The algorithms described in Chapter 3 were used to analyze three collections of music, each with matching sets of annotations. These collections are described in this section. Two, Beatles and RWC, are large and have been widely used in the literature; the other is based on music collected from the Internet Archive and was created as part of the present work.

### *4.2.1 The Beatles*

Between 1989 and 2000, musicologist Allan Pollack undertook the task of analyzing all the songs in the Beatles catalogue[15] (Pollack 2000). His analyses were comprehensive and included discussions of each song's style, form, melody, harmony, and arrangement, and all of his research was released for free on the Internet. A set of annotations for this corpus was created at the Universitat Pompeu Fabra (UPF) by adding timing information to Pollack's analyses; these annotations were in turn edited by

[15] < http://www.icce.rug.nl/~soundscapes/DATABASES/AWP/awp-notes_on.shtml> accessed 10 August 2010.

researchers at Tampere University of Technology (TUT). Both versions include 175 annotations and were released freely online under a Creative Commons license, although anyone wishing to use the corpus must naturally purchase the Beatles records themselves.

A third set of annotations for the Beatles catalogue, also distributed for free online, has been created at Queen Mary's Centre for Digital Music (CDM). These are also based on Pollack's research, but were created independently of the UPF and TUT sets and include all 180 songs on the Beatles' twelve studio albums. This includes five songs that were omitted from the UPF and TUT versions, presumably because each's formal structure was regarded as too idiosyncratic or ambiguous: the omitted songs include "Happiness is a Warm Gun," with ABCD structure, and the infamous "Revolution 9."

A breakdown of the vocabulary of both of these corpora of annotations are shown in Figure 4.4 and 4.5. The chart shows that over 80% of the section labels in each corpus are one of the basic five terms: "intro," "verse," "chorus," "bridge" and "outro." Most of the other labels are variations of these five, such as "verseA" and "chorusB," or

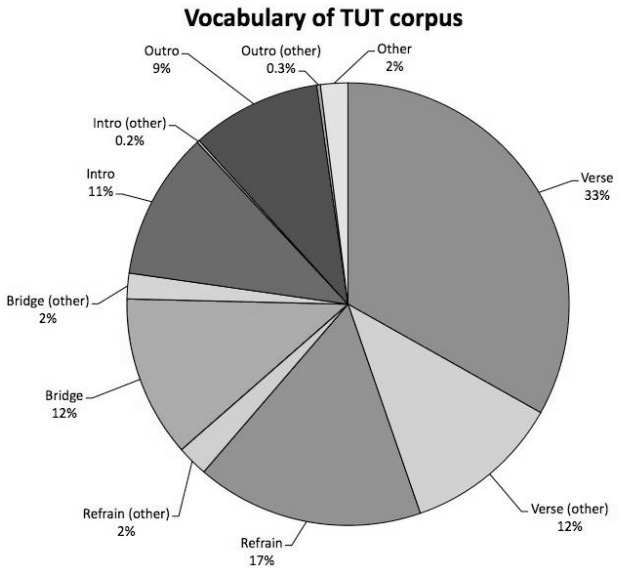



Figure 4.4. Vocabulary breakdown for the Tampere University of Technology (TUT) corpus of 175 annotations of Beatles songs. Each of the five most common labels is divided into two sections: the first tallies all verbatim appearances of the label, and the "other" one tallies instances where the label occurs in some varied form.

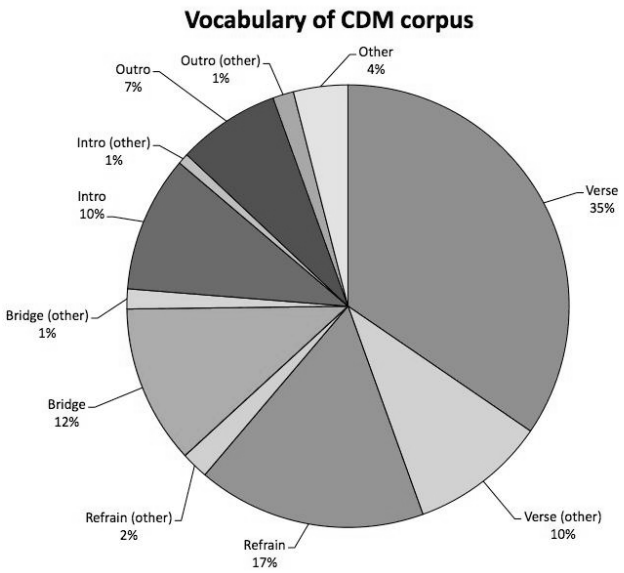



Figure 4.5. Vocabulary breakdown for the Center for Digital Music (CDM) corpus of 180 annotations of Beatles songs. Each of the five most common labels is divided into two sections: the first tallies all verbatim appearances of the label, and the "other" one tallies instances where the label occurs in some varied form.

"bridge_solo," and the remaining few belong to no obvious category at all, including "MR" and "break."

All 180 Beatles songs are studied in this thesis, and they are evaluated using both the TUT and the CDM sets of annotations. Due to the popularity of the Beatles and the availability of these annotations, this is one of the most widely used corpora for the evaluation of structural analysis algorithms, including the 2009 MIREX event. By including it in the present evaluation, these results will be directly comparable to many previous results.

### *4.2.2 Real World Computing*

In 2001, the Real World Computing (RWC) Partnership of Japan released a set of five music databases that were specifically designed for use in research (Goto et al. 2002). All of the music in these databases was recorded for RWC, and although the copyright for the music is retained by Japan's National Institute of Advanced Industrial Science and Technology (AIST), the music is made available at cost to research institutions. One subset of the RWC database is used in this thesis: the Popular Music

Database (PMD), comprising 80 songs in the style of Japanese popular music, and 20 songs in the style of Western popular music, by 35 artists in total. Note that none of the songs were actually released for public consumption as singles: they were commissioned exclusively for RWC.

RWC later released a set of annotations for each database, including structural annotations, which are freely available online (Goto 2006b). These annotations were created by a graduate student studying music. Although RWC includes jazz and classical corpora, their Popular Music Database has been used most frequently among structure analysis researchers. To be able to directly compare the results of the present evaluation with these previous ones, the PMD and its accompanying annotations were used in this thesis.

The RWC vocabulary is far more constrained than the TUT and CDM annotations, containing a total of just 14 unique labels. As with TUT and CDM, these consist of variations on five basic terms.

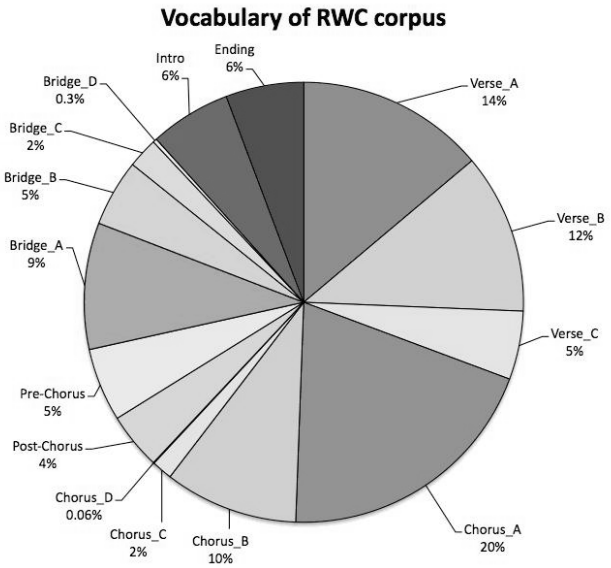



Figure 4.6. Vocabulary breakdown for the Real World Computing (RWC) corpus of 100 annotations of songs from the RWC Popular Music Database. Each of the fourteen different labels is tallied separately.

### *4.2.3 Internet Archive*

Two factors have motivated the development of a new corpus of music for this thesis: the general focus of available corpora on popular music, and the unfortunate expense of obtaining the music for which annotations are made. After outlining these factors in greater detail, this section describes the new corpus.

#### 4.2.3.1 Motivation for new corpus

Like the above collections, most other corpora of structural annotations that have been created focus on popular music, and likewise most of the algorithms discussed in Chapters 2 and 3 are designed to address popular music as well. This may be because popular music is regarded as generally having simpler and more obvious structure than classical, jazz, and other genres. Another reason may be that potential applications, such as retrieval and thumbnailing (discussed in Section 2.4), could perhaps be most lucratively applied to popular music collections. However, to more fully understand the robustness and applicability of these algorithms, one ought to evaluate them on corpora that include other genres of music.

In addition, it is desirable that musical corpora used for evaluation are shared between researchers so that algorithm performance may be more accurately compared. However, sharing music data is often illegal due to copyright restrictions. While annotations for the above two datasets are public, the collections of music they accompany are expensive to obtain. While this is the case for most popular music, there is a wealth of music legally available on the Internet that could form the basis of new, open musical corpora. One particularly large source of music is the Internet Archive, a huge online library that as of July 13th, 2010, contains over 577,000 items, including over 80,000 recordings of full concerts, and over 11,000 recordings from 78 RPM records and cylinders. Almost all of this audio is freely downloadable and redistributable, either because it has been released under a Creative Commons license, or because it is in the public domain.

#### 4.2.3.2 Description of new corpus

For this thesis, I have assembled a new collection of pieces of music and annotated their structure. The set contains 15 each of classical and jazz pieces

downloaded from the Internet Archive. The contents of each set, including instrumentation and, where known, composers and performers, are summarized in Tables 4.1 and 4.2.

In order to simplify the annotation process, and because the algorithms being evaluated in this thesis had never been tested on classical or jazz music before, the pieces in these corpora were selected because their large-scale structure was deemed to be especially unambiguous. The classical set thus consists mainly of minuets and middle movements from sonatas and symphonies, which are often modelled after traditional and straightforward dance forms. Likewise, the jazz set is drawn principally from early genres, such as ragtime and Dixieland, in which exact repetition was more common than in later styles.

| *Title* | *Composer* | *Instrumentation* |
|---|---|---|
| Piano Sonata no. 8 op. 13 ("Pathétique"), second movement | Ludwig van Beethoven | Piano |
| Sonata no. 14 ("Moonlight"), op. 27 no. 2, second movement | Ludwig van Beethoven | Piano |
| Symphony no. 8, op. 93, third movement | Ludwig van Beethoven | Symphony |
| String Trio in E-flat major, op. 34 no. 3, G 103 | Luigi Boccherini | String trio |
| Suite in D | George Handel | String quartet |
| String quartet no. 41 in D major ("The Frog"), op. 50 no. 6, third movement | Joseph Haydn | String quartet |
| Symphony no. 92 ("Oxford"), third movement | Joseph Haydn | Symphony |
| Symphony no. 40 in G minor, K. 550, third movement | Wolfgang Amadeus Mozart | Symphony |
| Minuet in G major, op. 14 no. 1 | Ignacy Jan Paderewski | Piano |
| Minuet in G minor | Christian Petzold | Horns |
| Gigue en Rondeau | Jean-Phillippe Rameau | Synthesized guitar |
| Symphony no. 1, third movement | Robert Schumann | Symphony |
| Symphony in E-flat major, op. 13 no. 3, third movement | Johann Stamitz | Chamber orchestra |
| Nutcracker Suite no. 2, "Marche" | Pyotr Tchaikovsky | Orchestra |
| Minuet | Unknown | Mandolin ensemble |

Table 4.1. Summary of Internet Archive: Classical database.

| *Title* | *Composer or Performer* | *Instrumentation* |
|---|---|---|
| Frankie & Johnny | Louis Armstrong | Singer and piano |
| Perfidia | Benny Goodman | Singer and full band |
| Hungarian Rag | Edison Military Band | Full band |
| Henpecked Blues | Isham Jones and His Orchestra | Full band |
| Joseph Lamb | Scott Joplin | Piano |
| Pine Apple Rag | Scott Joplin | Piano |
| The Ragtime Dance | Scott Joplin | Piano |
| Piano Roll Blues | Scott Joplin | Piano |
| Dixieland One-Step | Lopez and Hamilton's Kings of Harmony | Jazz combo |
| Maple Leaf Rag | Tom McDermott | Piano trio |
| Four Or Five Times | King Oliver | Singer and full band |
| Sensation Rag | Original Dixieland Jazz Band | Jazz combo |
| Eccentric Rag | Oriole Orchestra | Full band |
| Alexander, Where's That Band? | Parham-Pickett Apollo Syncopators | Full band |
| Say It With Music | Red Nichols & His Five Pennies | Singer and full band |

Table 4.2. Summary of Internet Archive: Jazz database.

The annotations were created using the Sonic Visualiser environment by the author, a graduate student with extensive musical training and a prior degree in music. Although the annotations must be taken as subjective as warned in Section 4.1.1, some important steps have been taken to mitigate this caveat: first, each annotation contains at least two hierarchical layers, maximizing the chance that an analysis that is correct at one level will be fairly evaluated; second, semantic labels (e.g., "minuet," "trio," and "da capo") were set aside in favour of neutral labels (i.e., "A," "B," "C," etc.) so that the annotations would reflect the organization of musical ideas rather than of musical functions.

### *4.2.4 Summary*

Three musical corpora including 310 pieces are studied in the present thesis: the Beatles catalogue of studio albums (180 pieces), the RWC Popular Music Database (100 pieces), and a new corpus of freely available classical and jazz pieces downloaded from the Internet Archive (30 pieces). The latter corpus was created to investigate how well the

algorithms of Chapter 3 perform on novel genres of music, and to establish a public and shareable database for structural analysis. Corresponding structural annotations for all corpora are publicly available.

## 4.3 Evaluating structural descriptions

The performance of an algorithm is evaluated by comparing the output of the algorithm to the ground truth annotation, produced as described in the previous section. However, structural descriptions are complex data and comparing two descriptions is not straightforward. Several evaluation metrics have been used and reported by various researchers, and this multitude of metrics is another reason why comparing the performance of different algorithms can be difficult. Since this situation was publicized within the community by Lukashevich (2008), greater awareness has in some cases led to more thorough reporting of evaluation metrics; however, disparities in evaluation reporting persist. To maximize the potential of the present evaluation to be cross-referenced with previous ones, most of the previously used metrics are also reported in this work. The meaning and derivation of these metrics are described in this section.

A structural analysis consists of two layers of information: a segmentation, and a grouping of the resulting sections. Evaluation metrics tend to consider each layer separately: some metrics consider only the accuracy of the boundaries, and other metrics consider only the accuracy of the labelling. Many metrics come in pairs in which one indicates if the piece has been under-segmented or if the estimation includes too little detail, and the other indicates if the piece has been over-segmented or if the estimation is over-detailed; often, a third measure will combine the two. The next section describes methods of evaluating an estimated segmentation, and the section afterward describes methods of evaluating an estimated labelling.

### *4.3.1 Boundary evaluation*

*Precision and Recall*

The annotated boundaries are a set of time points that analysis methods aim to recover precisely. Equivalently, this can be thought of as a classification task, where each audio frame is classified as a boundary or as a non-boundary. In either view, the quality

of the set of estimated boundaries can be evaluated using the familiar pair of evaluation metrics known as precision and recall. Recall is the percentage of the total number of annotated boundaries that were recovered, and precision is the percentage of estimated boundaries that were correct.

Given a set of estimated boundaries $E$ and a set of annotated boundaries $A$, the set of correctly retrieved boundaries is indicated by their intersection $A \bigcap E$ . The precision $p$ and recall $r$ may thus be calculated as:

$$p = \frac{|A \bigcap E|}{|E|}$$

$$r = \frac{|A \bigcap E|}{|A|}$$

Over-segmentation tends to decrease the precision, while under-segmentation will decrease the recall. The two are usually combined as an *f*-measure statistic, which is the harmonic mean of the two:

$$f = \frac{2pr}{(p + r)}$$

Each of the above three metrics takes a value between 0 and 1, with 1 representing perfect performance.

One aspect of the above definition that was left unspecified is how to decide if a boundary is correct. This amounts to setting the proximity in seconds that an estimated boundary must have to the nearest annotated boundary. The selection here is somewhat arbitrary, but the community has settled on two values: a 0.5-second margin and a 3-second margin, corresponding to windows with widths of 1 and 6 seconds around each boundary.

*Median boundary distance*

Because the above method relies on a fixed threshold, it does not distinguish a perfectly estimated boundary from a boundary that was just barely within the window — nor does it distinguish a boundary that was just outside the window from one that was 20 seconds away from the nearest annotated boundary. Two median boundary distance measures help paint a fuller picture by measuring how far each boundary was from the

nearest annotated boundary. The median true-to-guess distance finds the minimum distance from each annotated boundary to any estimated boundary, and returns the median of these values. This value is analogous to recall, since if the estimated description is highly under-segmented, then the median true-to-guess distance will be high. The median guess-to-true distance finds the median of the minimum distance from each estimated boundary to any annotated one. It is analogous to precision, and suffers in the case of an over-segmented description.

*Directional Hamming distance*

The directional Hamming distance $d_{AE}$ used by Abdallah et al. (2005) also evaluates the quality of the segmentation without explicit regard to the labelling. Effectively, it considers each segment of the estimation in turn and measures how well the single best-matching segment in the annotation matches. It is calculated by first identifying, for each estimated segment $e_j$, the annotated segment $a_{max}$ that maximally overlaps $e_j$. The spans of $e_j$ which overlap any segment $a_i$ other than $a_{max}$ is kept, and the sum of these spans over the set of estimated segments gives $d_{AE}$. Expressed mathematically, this gives:

$$d_{AE} = \sum_{e_j} \sum_{a_i \neq a_{\max}} \left| e_j \cap a_i \right|$$

Thus, $d_{AE}$ is a measure of under-segmentation, since its value will be high if boundaries have been missed and the estimated segments are longer than the annotated ones. The inverse directional Hamming distance $d_{EA}$, calculated by swapping the role of the annotation and estimation in the above calculation, likewise gives a measure of over-segmentation.

In the worst case scenario, $d_{AE}$ will tend toward the length of the whole piece, so Abdallah et al. (2005) normalized it by this length to produce a "missing rate" score denoted by $m$. This takes a value between 0 and 1, with 0 indicating perfect segmentation; so that perfect segmentation would give the value 1, Lukashevich (2008) reported the inverse score $1 - m$. In a similar manner, the inverse directional Hamming distance $d_{EA}$ can be normalized to give a "false alarm" or "fragmentation" score $1 - f$.

### *4.3.2 Grouping evaluation*

Evaluating the quality of a labelling scheme is more complex than evaluating the retrieved section boundaries: rather than to retrieve a set of items, the goal is to retrieve a set of relationships between items. Different perspectives on this problem have inspired different metrics to be used: some, such as the Rand index and pairwise precision and recall, spring from a clustering interpretation, asking, "Are the items that were clustered together in the annotation also clustered in the estimation?" Others, such as conditional entropy, originate from the perspective of information theory, and ask, "Is the information encoded in the annotation effectively restated in the estimation?"

#### 4.3.2.1 Clustering metrics

The task of music structure analysis has been frequently approached as a clustering problem (see Section 2.3), in which the task is to group together those portions of the piece of music which were grouped together in the ground truth. If the algorithm produces semantic labels, then evaluating the result is straightforward: as in Paulus and Klapuri (2009), one may simply compare the label estimated for each frame to that provided in the ground truth, and calculate the percentage of correctly labelled frames. However, most algorithms produce non-semantic output such as "AABBA," which should be compared to the ground truth in a permutation-invariant manner. That is, if the ground truth is "ABCBC," then the estimated description "ACBCB" should be given a perfect score. Rather than evaluate specific labels, then, the task is to evaluate the *relationships* between the labels. Three methods of doing so are described in this section: pairwise precision and recall, the Rand index, and the purity metric.

*Pairwise precision and recall*

Proposed by Levy and Sandler (2008), pairwise precision and recall consider the pairwise matches between frames in the annotation and the estimation. The set of all pairs of frames $(i, j)$ in the annotation whose labels $L_i$ and $L_j$ are a match is a set of grouping relationships that can be seen as a set of items to be retrieved, so that we may apply the familiar precision and recall metrics discussed in Section 4.3.1. Taking $M_A$ to be the set of all pairwise matches in the annotation and $M_E$ to be the set of pairwise matches in the estimation, we have again:

$$p = \frac{\left|M_A \cap M_E\right|}{\left|M_E\right|}$$

$$r = \frac{\left|M_A \cap M_E\right|}{\left|M_A\right|}$$

where $|x|$ counts the number of elements in the set $x$. The origin of these measures is depicted in Figure 4.7 as a Venn diagram: the ratio $|a|/(|a|+|c|)$ gives precision, while $|a|/(|a|+|b|)$ gives recall. The set of matching pairs could also be visualized as a binarized SSM where a white pixel at position $(i, j)$ corresponds to a pairwise match; viewed this way, pairwise precision and recall would rate how well these white pixels were retrieved. High pairwise recall and low precision indicates that the estimation was not conservative enough in judging the similarity of different regions; conversely, low recall and high precision indicate that the estimation was too fine, distinguishing as separate regions that ought to have been considered a whole.

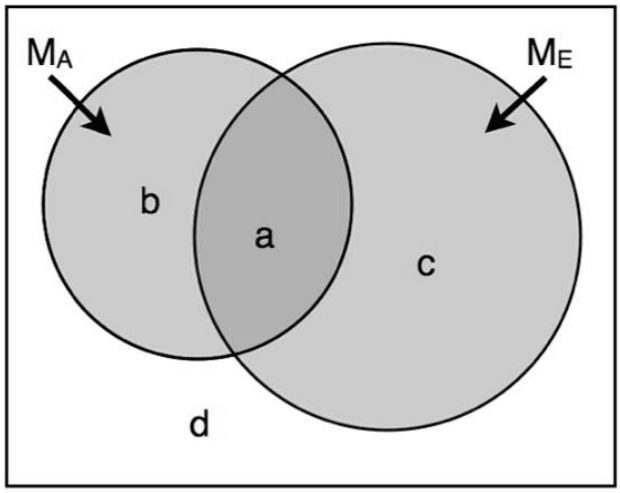



Figure 4.7. Venn diagram illustrating the calculation of pairwise precision and recall. $M_A$ represents the set of pairwise matches in the annotation, $M_E$ the pairwise matches in the estimation. The set *a* indicates the overlap between these sets, *b* the set of matches that are found in the annotation but not the estimation, *c* the set of matches in the estimation that are missing from the annotation, and *d* the set of pairs that do not match in either set. Precision may be calculated as the ratio |a|/(|a| + |c|), recall as |a|/(|a| + |b|), and the Rand index as (|a| + |d|) / (|a| + |b| + |c| + |d|)

*Rand index*

The Rand index was proposed as a structure evaluation metric for the 2009 MIREX event. It is related to the pairwise precision and retrieval, with a subtle adjustment: in addition to appraising pairwise matches, it appraises pairwise anti-matches as well. In other words, the Rand index counts as agreements pairs of frames that do not have matching labels in either the annotation or estimation. With respect to Figure 4.7, the formula for the Rand index can be expressed as:

$$R = \frac{|a| + |d|}{|a| + |b| + |c| + |d|}$$

Like the *f*-measure, the Rand index considers precision and recall together.

*Purity*

The speaker purity and cluster purity metrics were developed to evaluate a clustering task in which the goal was to group together utterances from the same speaker, as noted by Lukashevich (2008). Analogously, the goal in structural analysis is to group together those audio frames that originate from the same section-types in the annotation. We may thus borrow the average cluster purity (ACP) score, which measures how well regions that were grouped in the annotation have been grouped in the estimation, and its opposite, the average speaker purity (ASP), which assesses how well the distinctness of each label has been preserved. ACP and ASP are similar to pairwise precision and recall, respectively, except that the purity scores are penalized even more when a set of frames that should be all assigned to the same label (according to the annotation) are divided among several labels.

The steps for calculating the ACP are provided by Lukashevich (2008) and are outlined below. To calculate ASP, simply swap the role of the annotated and estimated values.

1. Consider all the audio frames associated with a particular label *j* in the annotation.
2. Consider the parallel audio frames in the estimation. They may be associated with a number of labels; tally up separately the number of frames associated with each label.
3. Calculate the sum of the squares of these tallies, and normalize by the total number of frames in the annotation with label *j*.

4. This gives the cluster purity of the label $j$. The average cluster purity is taken as the weighed sum of each label's purity.

The result will have a maximum when all of the estimated frames have been given the same label, and will reach a minimum when the frames in the estimation are evenly distributed among all the segment labels. Note that unlike pairwise precision, the lowest possible score for ACP is not 0, but 1 divided by the number of labels, a result achieved when the frames annotated with a particular label are uniformly scattered among all the labels in the estimation.

As with precision and recall, the ACP and ASP may be combined into a measure $K$, which is the square root of the product of the two purity scores.

#### 4.3.2.2 Information-theoretic metrics

One may ask abstractly, "How much of the information contained in the annotation is conveyed by the estimation?" The answer can be characterized using several related information-theoretic measures, including conditional entropy, mutual information, and Lukashevich's (2008) proposed over- and under-segmentation scores.

In information theory, the "quantity" of information in a set of values is estimated as the minimum number of bits it would take to encode the data, according to some encoding scheme. (Technically, the number of bits required to encode some set of data will change depending on the encoding scheme, and the optimal encoding scheme is not normally discoverable.) This number will increase whenever there is greater uncertainty, or entropy, in the data: for instance, the string $s_1$ = "AAABBB" would require fewer bits to encode than the string $s_2$ = "ABBABA," since the former is organized more simply. Its entropy $H(s_1)$ would likewise be smaller than $H(s_2)$.

*Conditional entropy*

The conditional entropy $H(s_2|s_1)$ between two sets of data is the amount of extra information required to encode the data set $s_2$ based on the set $s_1$. For example, if the data sets were strings, then the conditional entropies would be analogous to the string edit distance discussed in Section 2.3.2.3. $H(A|E)$ expresses the amount of information missing from the estimation, while $H(E|A)$ expresses the amount of spurious information contained in the estimation.

Conditional entropy may be calculated by treating the data sets as probability distribution functions and calculating the conditional probabilities between each pair of labels. For each label $a_i$ in the annotation that occurs with probability $p(a_i)$ (i.e., that accounts for a proportion $p(a_i)$ of the annotation) we calculate the conditional probability $p(e_j|a_i)$ that an audio frame with label $a_i$ was given the label $e_j$ in the estimation. The following equation gives the conditional entropy $H(E|A)$ in terms of these values:

$$H(E|A) = -\sum_i p(a_i) \sum_j p(e_j|a_i) \cdot \log_2 p(e_j|a_i)$$

*Over- and under-segmentation scores*

The minimum conditional entropy is 0, but because the amount of information in each piece of music's annotation is different, the maximum value is also different in each case. Lukashevich (2008) proposed normalizing the conditional entropy $H(E|A)$ by its estimated maximum value. As with the purity metric, this maximum occurs when all frames in the annotation are equally likely to be assigned to any label used in the estimation. By substituting the value $p(e_j|a_i) = 1/N_e$, Lukashevich obtains a maximum conditional entropy, and uses these to create an over-segmentation score $S_o$. Normalizing by $N_a$, the number of labels in the annotation, gives an under-segmentation score $S_u$.

$$S_o = 1 - \frac{H(E|A)}{\log_2 N_e}$$

$$S_u = 1 - \frac{H(A|E)}{\log_2 N_a}$$

Each of these scores takes values between 0 and 1, with perfect annotations garnering a 1. Lower over-segmentation scores indicate the presence of spurious information in the estimation, while lower under-segmentation scores suggest that the estimation is missing information from the annotation.

*Mutual information*

The mutual information $I(X, Y)$ between two data sets is the amount of information they have in common, and may be derived intuitively from the conditional entropy. If the conditional entropy $H(A|E)$ expresses the amount of annotated information that is not contained in the estimation, and the entropy $H(A)$ expresses how much

information is in the annotation to begin with, then their difference gives the amount of information shared by the annotation and the estimation. Put simply:

$$I(A,E) = H(A) - H(A|E)$$

Mutual information may also be calculated from the similarity of the probability distributions that define the two descriptions, using the following equation:

$$I(A,E) = \sum_i \sum_j p(a_i|e_j) \cdot p(e_j) \cdot \log\left(\frac{p(a_i|e_j)}{p(a_i)}\right)$$

When the distribution of two labels $a_i$ and $e_j$ are similar, the argument of the above sum will be large and the mutual information greater. However, like the conditional entropy, the magnitude of $I$ is not normalized and depends on the amount of information contained in the annotation.

## 4.4 Summary

This chapter began with a discussion of important issues relating to the production of structural annotations. Fuzzy notions of musical similarity and the hierarchical organization of music were noted as fundamental obstacles that some researchers are nevertheless beginning to overcome. The three corpora of music and accompanying annotations to be analyzed in the present work were described; they include one new corpus that consists of publicly available music and that may therefore be shared by researchers in the future. Finally, the many metrics used to perform the present evaluation were defined. These include most of those used in previous evaluations, including all of those used at MIREX 2009.

# 5 Results and Discussion

An evaluation was performed using the three corpora of music described in Section 4.2: the Beatles catalogue, the Real World Computing (RWC) Popular Music Database, and the set of pieces from the Internet Archive. Each song in each corpus was analyzed by the three structure analysis algorithms (Barrington et al. 2009, Levy and Sandler 2008, and Peiszer 2007) and by the two segmentation algorithms (the Echo Nest and Mestres 2007) described in Chapter 3. The output of each algorithm was compared to the corresponding annotated descriptions in each of the various ground truth sources described in Section 4.2: the Beatles annotations provided by the Centre for Digital Music (CDM) and by the Tampere University of Technology (TUT); the RWC annotations provided by RWC; and the hierarchical Internet Archive annotations created for this thesis, including one larger-scale and one smaller-scale description of each piece. The performance of each algorithm on each song was measured using the evaluation metrics defined in Section 4.3, which appraise the estimated boundaries and the estimated labels of each description.

This chapter summarizes the methodology and the results of this evaluation. The first section describes how the evaluation was implemented. It includes a summary of the input parameters and of the performance metrics used in the evaluation, as well as discussions on how the ground truth files were handled and on which performance baselines were used.

The second section presents the results of this evaluation. Because there are so many tunable parameters in the algorithms and different versions of the ground truth, the resulting collection of evaluation data is quite large: over 2 million metrics were calculated. Section 5.2 presents an overview of these results, presenting and discussing several views on the data. The full results and raw output of each algorithm are available online.[16]

[16] <http://www.music.mcgill.ca/~jordan/structure>

## 5.1 Review of evaluation procedure

The algorithm parameters and evaluation metrics used in this thesis are described in detail in Chapters 3 and 4, respectively, but are briefly recalled in this section, with the relevant information summarized in tables. This section also discusses a handful of ways in which the ground truth data may be manipulated before being used for evaluation, and a description of some naïve structure analysis strategies that are implemented as baselines for this evaluation.

### *5.1.1 Summary of algorithm parameters*

Five analysis algorithms were tested: Barrington et al. (2009), Levy and Sandler (2008), Peiszer (2007), the Echo Nest, and Mestres (2007). Of these, the latter two provided only segmentation data and did not require any input parameters to be specified. The input parameters specified for the first three included the feature set to be used and *k*, the number of clusters (i.e., section label types) that each algorithm would attempt to model. Levy and Sandler's algorithm also required one to choose a value for *N*, the minimum segment length. As well, three different versions of the clustering stage of Peiszer's algorithm were implemented, although each version's segmentation of a particular song were identical to each other.

Although Barrington et al.'s algorithm did not require setting any parameters beside the choice of feature and the value for *k,* this evaluation did consider both the coarse and the fine-grain output of the algorithm. The fine-grain description is the final output of the algorithm, but usually does not differ substantially from the coarse description that the algorithm creates partway through. Since Barrington et al.'s algorithm takes quite a long time to operate, sometimes taking several hours to compute a single description, it would be valuable to know if the quickly-found coarse descriptions are just as good as the fine-grain ones.

The parameters used in this experiment are summarized in Table 5.1.

| Algorithm | Features | Number of cluster types *k* | Other parameters | Number of parameter combinations |
|---|---|---|---|---|
| Barrington et al. | chroma, MFCC | 3, 4, 6 | fine or coarse output | 6 |
| Levy and Sandler | CQT, MFCC | 3, 4, 5, 6, 8, 10 | minimum section length: 4, 10 seconds | 24 |
| Peiszer | CQT, MFCC | 3, 4, 5, 6, 8, 10 | clustering approach: k-means, agglomerative, DTW | 36 |

Table 5.1. Summary of input parameters used for each algorithm. Note that fewer values for *k* were used with Barrington et al.'s algorithm since it required substantially longer to run.

### *5.1.2 Summary of ground truth variations*

Each algorithm, after analyzing a piece of music, provides a segmentation of the piece and a grouping of these segments. This grouping is symbolic, rather than semantic; that is, the estimated section labels are simply "1," "2," "3," and so forth, rather than "verse," "chorus," or "bridge," as used in most of the ground truth. The semantic content of these ground truth labels is typically ignored in evaluation—certainly it must be when none of the algorithms estimate semantic labels—but the process of reducing semantic labels to simple groupings demands certain choices.

For instance, in the RWC ground truth files, individual sections are normally split up into several parts, provided with labels such as "verse_a" and "verse_b." In the evaluation step, one may either wish to group these sections together, since they are all part of the "verse," or to consider them as distinct labels. The CDM and TUT ground truth sets also include labels such as "verseS" or "verse_(guitar_solo)," both of which indicate a version of the verse featuring a solo. Similarly, some of the annotations provided in this thesis for the Internet Archive corpus include prime markers to distinguish variations: e.g., A' indicates a variation on the section A. Deciding whether or not sections with related labels should be grouped together is a subjective one, as both interpretations have musicological merit. Thus, in the present evaluation, both interpretations were used: new versions of the ground truth files were produced that

conflated and grouped together similar terms, and the output of the algorithms was evaluated against both versions.

In their evaluation using RWC data, Barrington et al. (2009) went further in constraining their label vocabulary: they mapped each section to one of four labels, "verse," "chorus," "bridge," and "other," and merged neighbouring sections with the same label. For the present evaluation, a third version of the ground truth for the RWC, CDM, and TUT corpora was produced according to this procedure. Since the Internet Archive annotations do not include any semantic information, there is no obvious way to produce a four-label version of this corpus, so only the segment-merging aspect of Barrington's procedure was applied to this corpus. Note that this procedure potentially misrepresents the data, since it may be unlikely that the "intro," "pre-chorus," and "ending" sections of a song actually sound alike, although they are all given the label "other." Secondly, it is usually considered important to detect boundaries even between similar segments, although these are not part of this ground truth.

In total, then, three versions of the ground truth were included. The first was the original version, treating each distinct label as distinct. The second conflated similar terms, discarding the label prefixes and suffixes that the annotators used to indicate nuanced differences between sections. The third was a four-label version in which all sections not belonging to one of the three main song functions (verse, chorus, and bridge) were grouped together, and where neighbouring segments with the same label were merged; for the Internet Archive corpus, only this merging step was applied. To illustrate the difference between these versions, the original, conflated, and four-label versions of the ground truth for the Beatles song "I Am The Walrus" is shown in Figure 5.1.

### *5.1.3 Summary of baseline estimations*

When evaluating the performance of an algorithm, it is essential not only to compare different algorithms' performance to each other, but also to the performance of a random or naïve baseline procedure. Ensuring that algorithms perform significantly better than chance is important both to validate the algorithm and the evaluation procedure.

Different authors have used different baseline analysis methods. In their evaluation of segmentation algorithms, Turnbull et al. (2007) used two baselines, both of

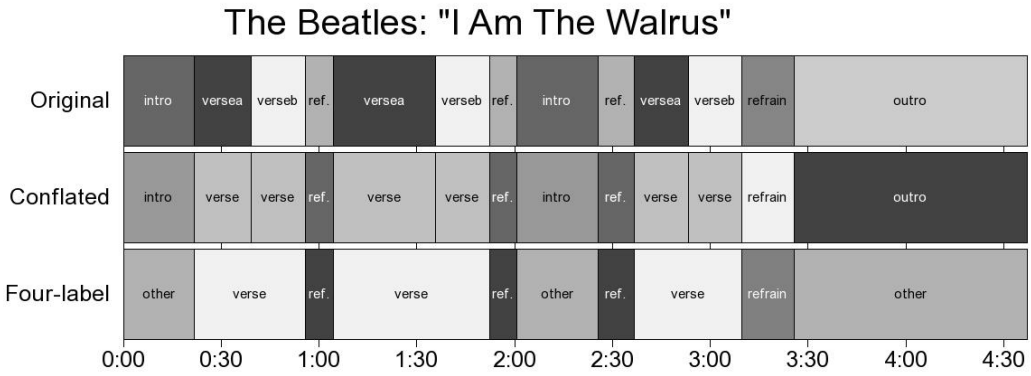



Figure 5.1. Illustration of difference between different versions of the same ground truth file: the CDM annotation of "I Am The Walrus."

which estimated 10 boundaries per song: the first baseline spaced these boundaries uniformly throughout the piece, and the second placed them randomly. Peiszer (2007) used a baseline segmentation method where each segment was the same length $L$, with the first segment being placed $L/2$ seconds from the beginning of the track. He used values of $L$ of 10, 15, 20, and 30 seconds.

To evaluate how well the segment-labelling step of his algorithm had performed, independent of its success at the segmentation step, Peiszer used a baseline in which segments determined by the ground truth boundaries were labelled randomly. The maximum possible number of segment types or different labels used was determined by whatever value for $k$ was chosen for the algorithm. Chai (2005) used a constant baseline in which the entire song was labelled as a single section. Barrington et al. (2009) did the same, and included a random baseline in which each window of audio was given a random label.

All of these methods were appropriated for this thesis, a total of eight approaches to boundary estimation and two approaches to labelling. The boundary estimation methods are summarized in Table 5.2. For each one, two baseline descriptions were created: in the first, each segment was given the same label; in the second, each segment was given one of $k$ random labels. This parameter $k$ was varied in the same manner as for the analysis algorithms.

| Baseline segmentation | Description |
|---|---|
| Constant | No segment boundaries |
| 10E | 10 evenly-spaced boundaries |
| 10R | 10 randomly-placed boundaries |
| L10 | Boundaries spaced L seconds apart, beginning at time L/2 |
| L15 | |
| L20 | |
| L30 | |
| Framewise | Segment boundaries placed every second |

Table 5.2. Summary of baseline segmentation methods employed. For each method, two labellings were provided: one giving each section the same label, the other giving each section a random label.

### *5.1.4 Summary of evaluation metrics*

Several evaluation metrics have been used to assess the accuracy of the boundaries and segment labels of an estimated structural description. They are described in detail in Section 4.3, and are summarized in Tables 5.3 and 5.4. Each metric is listed along with its best and worst values. In addition, it is noted what type of errors the metric is sensitive to: boundary-related metrics may reflect over-segmentation errors, under-segmentation errors, or both; likewise, labelling-related metrics may punish spurious information, or missing information, or either.

In general, the discussion in this chapter will focus mainly on the metrics that are normalized and that reflect both types of errors. These include: the two boundary *f*-measures with different thresholds, which balances the precision and recall with which the correct boundaries were retrieved; the pairwise *f*-measure, which describes how well the set of pairs of frames with the same label in the ground truth were retrieved; the combined cluster and speaker purity measure *K*, which estimates how well the section types in the ground truth were preserved by the estimated sections, and how efficiently the estimated sections accounted for them; and the Rand index, which measures the rate of agreement between all pairs of frames in each description.

| Evaluation metric | Best value | Worst value | Poor values indicate |
|---|---|---|---|
| Boundary precision with 1-second window B_P1 | 1 | 0 | Over-segmentation |
| Boundary recall with 1-second window B_R1 | 1 | 0 | Under-segmentation |
| Boundary *f*-measure with 1-second window B_F1 | 1 | 0 | Either |
| Boundary precision with 6-second window B_P6 | 1 | 0 | Over-segmentation |
| Boundary recall with 6-second window B_R6 | 1 | 0 | Under-segmentation |
| Boundary *f*-measure with 6-second window B_F6 | 1 | 0 | Either |
| Median true-to-guess distance MTG | 0 | - | Over-segmentation |
| Median guess-to-true distance MGT | 0 | - | Under-segmentation |
| Fragmentation rate 1-f (based on inverse directional Hamming distance) | 1 | > 0 | Over-segmentation |
| Missed boundaries rate 1-*m* (based on directional Hamming distance) | 1 | > 0 | Under-segmentation |

Table 5.3. Summary of evaluation metrics for boundary estimation.

## 5.2 Summary of results

As stated before, the volume of evaluation data collected for this work is great, and it would be unhelpful to display them in full here, although they are available in their entirety online.[17] Because of the high number of varying dimensions in this evaluation (including the choice of corpus, of ground truth, and of algorithm parameters), the data also elude immediate summary: it would be hasty, for instance, to average the results across the different corpora. This section therefore proceeds by examining and discussing different narrow views of the data. The following section examines how each algorithm's

[17] <http://www.music.mcgill.ca/~jordan/structure>

| Evaluation metric | Best value | Worst value | Poor values indicate |
|---|---|---|---|
| Pairwise precision $PW_P$ | 1 | 0 | Spurious information |
| Pairwise recall $PW_R$ | 1 | 0 | Missing information |
| Pairwise *f*-measure $PW_F$ | 1 | 0 | Either |
| Rand index | 1 | 0 | Either |
| Average cluster purity ACP | 1 | 1/*k* | Spurious information |
| Average speaker purity ASP | 1 | 1/*k* | Missing information |
| Combined cluster and speaker purity measure *K* | 1 | 1/*k* | Either |
| Conditional entropy H(E\|A) | 0 | H(E) | Spurious information |
| Conditional entropy H(A\|E) | 0 | H(A) | Missing information |
| Over-segmentation $S_O$ | 1 | 0 | Spurious information |
| Under-segmentation $S_U$ | 1 | 0 | Missing information |
| Mutual information I(A, E) | max(H(A), H(E)) | 0 | Either |

Table 5.4. Summary of evaluation metrics for label evaluation.

performance depended on the choice of corpus and ground truth; the section afterward considers the choice of the number of cluster labels *k* and other parameters. Because of the high number of data plots, all are found in the Appendix.

### *5.2.1 Choice of corpus and ground truth*

This first view on the evaluation data will illustrate how the performance of each algorithm depended on the choice of corpus and on what processing has been applied to the ground truth annotations. Figure A.1 contains five subfigures, one for each corpus of annotations, and displays for each type of ground truth the results the pairwise *f*-measure.

Each mean represents a number of data points equal to the size of the corpus. Figures A.2 and A.3 similarly display the results for the two labelling metrics, the purity metric *K*, and the Rand index, and Figure A.4 shows the results for two versions of the boundary-estimation *f*-measure: one with a 0.5-second tolerance and the other with a 3-second tolerance. All of these metrics assess the overall quality of the estimated analysis rather than just the over- or under-segmentation.

Each figure displays the best average result achieved by any baseline and any version of each algorithm: that is, given a particular author's algorithm, every combination of parameters was tried and the most successful parameter set for a particular metric was used to generate the result seen in the figure. Note that individual figures often report results from different versions of a single algorithm: for instance, on the original version of the CDM ground truth, the version of Peiszer's algorithm with the best pairwise *f*-measure had the number of clusters *k* set to 4 and used *k*-means clustering (see Figure A.2a), whereas on the conflated version of the ground truth, Peiszer's best result used three cluster labels and agglomerative clustering. The error bars in all figures indicate the standard error, so the visual observations that follow may not carry statistical significance.

#### 5.2.1.1 Labelling performance

In most cases, the best baseline method performed worse at labelling than the evaluated algorithms. This is in line with previous evaluations, such as Peiszer (2007) and Barrington et al. (2009). However, when considering the Beatles annotations (either the CDM or the TUT versions) and the 4-label versions of the ground truth files, some baseline methods performed as well as or even better than the best algorithms (e.g., see Figure A.3a, where one baseline method earned a *K* measure that was substantially higher than for any version of any algorithm). The reason for this may be that restricting the CDM and TUT ground truth files to 4 labels eliminates too much detail in certain songs, to the point that a degenerate approach that labels all regions as identical becomes a more reliable method than any algorithm. (Indeed, this was the best-performing baseline on both Beatles corpora.) This is certainly true of a few songs with unconventional annotations that do not use the base terms "verse," "bridge," or "refrain." For instance,

the song “Revolution 9” is annotated in CDM simply as “beginning,” “middle,” and “end,” so that the 4-label version is simply one long “other” section.

The surprising success of the baseline on Beatles songs is also apparent with the conflated version of the ground truth. In this version, sections such as “verse_(solo)” are grouped with other verses, and this leads to a similar degeneracy for many of the Beatles’ blues pieces, where each section is annotated as some variation of the same 12-bar pattern. See for instance the ground truth versions for the piece “Everybody’s Trying To Be My Baby” in Figure 5.2, devoid of detail. In cases like this, a baseline estimate will wind up being more reliably accurate than any analysis.

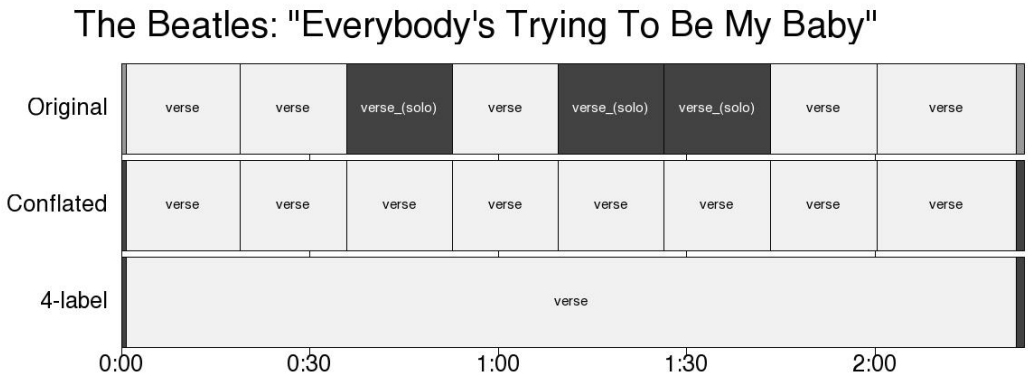



Figure 5.2. Illustration of different versions of ground truth for “Everybody’s Trying To Be My Baby.” Note that the 4-label ground truth contains virtually no information and no longer resembles a valuable analysis.

It is worth noting that the success of these naïve baselines seems mainly restricted to the Beatles annotations. As seen in Figures A.1–A.3, the baselines generally scored much lower with the Internet Archive corpus (subfigures c-d) and with the RWC corpus (subfigure e), which is composed almost entirely of variations on the base terms “verse,” “chorus,” and “bridge,” and for which Barrington et al. originally proposed using a 4-label version of the ground truth.

One exception to this observation is that the baseline performed just as well as Barrington et al.’s algorithm when evaluated with the Rand index. In all corpora and with all versions of the ground truth, the baseline with the highest Rand index always used

random labelling with the maximum number of clusters, using either the 15-second or the random segmentation method.

#### 5.2.1.2 Segmentation performance

As seen in Figure A.4a-j, the boundary estimation *f*-measure of most algorithms was generally poorer when the 4-label ground truth was used instead of the original ground truth, although this had little effect on the relative ranking between algorithms. (Note that the original and conflated ground truth had identical segment boundaries.) This could indicate that reducing the ground truth to four labels eliminates more true boundaries than spurious ones. Interestingly, the relative ranking of the algorithms, including the baseline, did not appear to depend on which ground truth version was used for any particular corpus. By contrast, the relative ranking did often change when the boundary detection threshold was changed; for instance, Barrington et al. was the most accurate on the CDM corpus with a 0.5-second threshold (Figure A.4a), but Peiszer was best with a 3-second threshold (Figure A.4b).

In all cases, the best performing baseline was the one that estimated one boundary per second. The only case where this baseline greatly outperformed the algorithms was with the small-scale annotations for the Internet Archive corpus (Figure A.4g-h). This makes sense, since it is the set of annotations with the highest number of segments: its average segment length is roughly 8 seconds, compared to roughly 20 seconds for each other set of annotations. (See Tables 5.5–5.7 for a comparison of the average number of sections, average section length, and average number of unique section labels in both versions of the ground truth.) Since the three-second tolerance for correct boundaries corresponds to a window around each boundary of 6 seconds, as many as three quarters (6 seconds out of 8 seconds) of randomly guessed boundaries may be judged correct.

This points to a flaw in the boundary evaluation process, since several estimated boundaries should not all be judged correct if they all target the same boundary in the ground truth. In future evaluations, a different procedure could be used where for each annotated boundary, only the nearest estimated boundary is counted as correct, with all extra nearby ones being counted as misses. With this procedure, a baseline that estimates

| Average number of sections | Original ground truth | 4-label ground truth |
|---|---|---|
| CDM | 10.16 | 7.33 |
| TUT | 9.21 | 7.34 |
| RWC | 17.11 | 11.03 |
| IA large-scale | 11.73 | 9.47 |
| IA small-scale | 31.87 | 28.80 |

Table 5.5. List of the average number of sections per piece of music in each of the five sets of annotations.

| Average length of sections | Original ground truth | 4-label ground truth |
|---|---|---|
| CDM | 16.31 | 19.65 |
| TUT | 15.09 | 19.33 |
| RWC | 13.54 | 20.87 |
| IA large-scale | 19.16 | 23.84 |
| IA small-scale | 7.26 | 8.33 |

Table 5.6. List of the average length of the sections in each of the five sets of annotations.

| Average number of section types | Original ground truth | Conflated ground truth | 4-label ground truth |
|---|---|---|---|
| CDM | 5.57 | 5.58 | 3.93 |
| TUT | 5.28 | 4.68 | 3.97 |
| RWC | 9.13 | 4.94 | 4.93 |
| IA large-scale | 5.47 | 6.23 | 6.23 |
| IA small-scale | 11.90 | 11.50 | 11.50 |

Table 5.7. List of the average number of unique section labels per piece in each of the five sets of annotations.

one boundary per second would still have a high recall rate, but its precision would drop sharply, appropriately reducing its *f*-measure.

The preceding analysis of the best achieved results for each algorithm as a function of which corpus and which ground truth version were used suggests that considering the conflated and 4-label versions of the annotations may be unnecessary, for several reasons. First, for the Beatles corpora, the conflated and 4-label versions of the ground truth appear to be sufficiently degraded versions of the original ground truth such that a naïve baseline labelling all regions as identical was evaluated positively. Second, for the RWC and Internet Archive corpora, the choice of which ground truth version was used did not appear to affect the measured success of the estimated labels. Finally, while most algorithms' success at estimating boundaries was affected by the choice of ground truth, the relative ranking of algorithms by their boundary estimation *f*-measure was not. Therefore, in the analysis and discussion of the results that follows, only the original version of the ground truth is considered.

### *5.2.2 Choice of algorithm parameters*

The next important result to examine is how the choice of input parameters affected each algorithm's performance. Aside from the Echo Nest and Mestres' systems, each author's system (as well as the baseline) required one to pick from a handful of input parameters. These parameters, summarized earlier in Table 5.1, included the choice of features, the number of cluster label types *k*, and others particular to each system.

#### 5.2.2.1 Number of cluster label types

We first consider the effect of the input parameter *k* on the success of each system. Figures A.5–A.8 show, for each corpus, how well each author's system performed on average, according to the same set of metrics considered earlier. Only the original version of the ground truth is considered here, but each data point in the figure is obtained by averaging together the results from every set of input parameters; for instance, the displayed values for Peiszer's algorithm average together the results obtained using both features (CQT and MFCCs) and all three clustering methods (*k*-means, agglomerative, and DTW). While this view therefore obscures the fact that one

system may have one parameter set that outperforms the others, it does give a fair impression of how the performance of each system varies with the input parameter *k*.

Again, the results paint a different picture for each corpus. With RWC and the small-scale version of the Internet Archive corpus, using a higher value for *k* nearly always led to improvements in performance in all metrics: witness the rising lines in Figures A.5d-e, A.6d-e, and A.7d-e. For Barrington et al. and Levy and Sandler, where the number of clusters *k* affected the segmentation, the boundary *f*-measure also improved slightly with greater values for *k* (see Figure A.8d-e). This conclusion is of course tentative for Barrington et al.'s system, since it was executed with only three values for *k*, due to its substantially longer runtime.

The same could not be said for the large-scale Internet Archive annotations or the Beatles annotations, where changing *k* did not have a consistent effect on the results: performance was static over each *k* for both boundary *f*-measures (Figure A.8a-c), while the Rand index rose for each algorithm with increasing *k* (Figure A.7a-c). The performance of Peiszer's algorithms fell sharply with increasing *k* for the pairwise *f*-measure (Figure A.5a-c) and purity metric *K* (Figure A.6a-c), although this trend was less pronounced with the other algorithms.

These disparate results make better sense when one considers the average number of segment types in each set of annotations (see Table 5.7). With RWC and the small-scale Internet Archive annotations, the true value for *k* was roughly 10, and setting *k* closer to 10 led to improved performance. On the other hand, the true value for *k* was roughly 5.5 for the Beatles and the large-scale Internet Archive annotations, so none of the tested values of *k* (which ranged from 3 to 10) were all that far from the correct value. Unsurprisingly, it appears to be true that setting *k* very far from the average number of segment labels in each corpus will hinder performance on that corpus.

It is interesting to note that among the four metrics plotted in Figures A.5–A.8, the Rand index obtained by each algorithm appears to increase monotonically with increasing *k* (see Figure A.7a-e). This was even true for the Beatles corpora (Figure A.7a-b), where the pairwise *f*-measure tended to fall with increasing *k* (see Figure A.5a-b). If the Rand index favours a higher number of cluster types, even more than are in the ground truth, it could be that the Rand index penalizes missing information more severely

than it penalizes spurious information. This certainly seems to be the case when compared to the pairwise *f*-measure: recall from Section 4.3.2.1 that both of these metrics consider how many pairs of frames with matching labels were correctly estimated, but only the Rand index measures how many pairs of frames with different labels were correctly estimated.

Among the structure analysis systems, Levy and Sandler's stands out, consistently outperforming the others, including the baseline; this was true in all corpora and for each evaluation metric (Figures A.5–A.7). However, recall that these figures plot the average performance over all sets of algorithm parameters; Figures A.1–A.3 showed that there was usually little or no difference in performance between the best-performing version of Levy and Sandler's and Peiszer's algorithms.

#### 5.2.2.2 Algorithm parameters

Finally, the effect of each algorithm's input parameters is investigated. Figures A.9–A.11 show, for each algorithm, the effect of applying different combinations of input parameters when evaluated on different corpora. These results were obtained using the original version of the ground truth with *k* set to 6 cluster labels. While this was not the best possible value for *k* to choose for each algorithm and each corpus, it represents a happy medium which, based on Figures A.5–A.8, did not lead to especially poor performance in any case.

*Barrington et al.*

As seen in Figure A.9a-c, Barrington et al.'s algorithm performed best using MFCCs as features when analyzing either the Beatles or the RWC corpora, and performed best using chroma as features when analyzing the Internet Archive corpus. This reflects what one might have expected based on some broad differences between popular music and classical or jazz music. The songs in the Beatles and RWC collections are mostly in the popular genre, where sections tend to stay in a single key area, and where different sections may feature different combinations of singers, different drum beats, or different instruments appearing in the foreground. On the other hand, the majority of the pieces in the Internet Archive corpus are mono-timbral, including several string quartets, piano rags, and Dixieland blues pieces. At the same time, tonal

relationships and melodic motifs can be more highly varied and structurally significant in classical music than in popular music, where it is not uncommon for a single song to use fewer than five chords.

There was not generally an important difference between the algorithm's coarse- and fine-scale outputs when using chroma features, but with MFCCs, the pairwise *f*-measure and the purity score *K* were both a small amount higher for the fine output (see Figure A.9a–b). It should be noted that the section label information estimated by Barrington et al.'s algorithm derives principally from its coarse analysis step, which estimates clusters of long-term audio sequences, and the time-consuming fine-scale analysis is only used to update the boundary locations. Since the coarse analysis is quite accurate on its own, one may consider applying a different boundary estimation technique; indeed, most of the other algorithms evaluated in this work would suffice, based on their higher boundary *f*-measures shown in Figure A.8.

*Levy and Sandler*

In contrast to Barrington et al.'s algorithm, the most important parameter for Levy and Sandler's appears to be the minimum segment length *N* (see Figure A.10a-c). For all corpora and for all the reported metrics, the difference between using the constant Q-transform (CQT) or MFCCs as features was small or non-existent. On the other hand, for nearly all corpora, using $N = 10$ seconds compared to $N = 4$ seconds led to greater pairwise *f*-measure and purity score *K* (the Rand index appears not to have differentiated between the quality of any of the used parameter sets; see Figure A.10c). This makes sense, since the average length of segments in most corpora was greater than 10 seconds (see Table 5.6). The single exception to this rule, where there was no difference in performance when using either value of *N,* was for the small-scale version of the Internet Archive corpus, which is the corpus of annotations with the smallest average segment length of 7.26 seconds.

*Peiszer*

Although the variation in performance was quite high between different parameter sets, the better feature to use for Peiszer's algorithm was always MFCCs, no matter which clustering algorithm was used (see Figure A.11a-c). Unlike Barrington et al. and Levy and Sandler, Peiszer uses all 40 Mel-frequency cepstrum coefficients. Normally

only the first 13 are kept, since these correlate most strongly with timbre, and the higher coefficients exhibit some dependence on the frequency content of the signal. However, in this application it is not necessary to avoid pitch information, so Peiszer's use of MFCCs may thus efficiently capture some information about both features, leading it to consistently outperform the CQT features. Peiszer (2007) does note that his system performed better when using all 40 MFCCs rather than 30 or the usual 13. Incidentally, Peiszer's pitch feature consists of an unwrapped chromogram, which itself indirectly provides a modicum of information about the spectral envelope.

Among the various clustering approaches, it appears that simpler is better: the *k*-means approach was most often the best, while the dynamic time warping (DTW)-based clustering method quite consistently underperformed compared to the others. However, the importance of the clustering method used was less important than the choice of feature: using the DTW approach and MFCC features led to greater pairwise *f*-measure and purity score *K* than using *k*-means clustering and CQT features.

One notable aspect of Peiszer's algorithm was that the best choice of features did not depend on the corpus under consideration. With Barrington et al., the best feature depended on the genre of music, and with Levy and Sandler, it seems that the average segment duration in the ground truth is important to know before setting the minimum segment duration *N*. On the other hand, no matter which corpus is being considered, the best choice of parameters for Peiszer's algorithm are MFCCs and *k*-means clustering.

### *5.2.3 Comparison with previous evaluations*

Most previous evaluations, including Abdallah et al. (2005 and 2006), Chai (2005), Lu et al. (2004), Peiszer (2007), and many others, are incompatible with the present evaluation either because they used different corpora of music or different evaluation schemes. The MIREX 2009 Structural Analysis task results are also unable to be compared to these, for although they include the 180 Beatles songs, additional songs were also tested and the Beatles results are not separable from them. Still, a fair number of recent evaluations have used the Beatles corpus (Paulus and Klapuri 2009) or subsets of it (Levy et al. 2007), or the RWC corpus (Paulus and Klapuri 2009; Turnbull et al. 2007; Barrington et al. 2009 and 2010).

In Tables 5.8–5.15, results from these studies are presented with comparable data from the present evaluation. In each case, the results obtained by the best-performing algorithm on the relevant corpus are presented, and the specific algorithm parameters are noted for each algorithm. Note that several groups of these tables (including 5.8–5.10; 5.12 and 5.14; 5.13 and 5.15) could have been combined if not for the fact that the studies considered included a different corpus, a different version of the ground truth, or a different metric. The resulting confusion is a reminder of why it is important to agree on standardized collections, annotations, and performance metrics.

The results generally show that the algorithms evaluated currently are competitive with those that were evaluated previously. On the RWC corpus, Peiszer obtained the best boundary *f*-measure of 0.680 (Table 5.14), and Paulus and Klapuri obtaining the best pairwise *f*-measure of 0.637 (Table 5.15). The same ranking was obtained on the Beatles corpus, where Paulus and Klapuri's (2009) algorithm obtained the best pairwise *f*-measure (Table 5.11), although Barrington et al. (2009), Levy and Sandler (2008), and Peiszer (2007) were all within 3% of this score. Peiszer's algorithm obtained the best boundary estimation *f*-measure by a somewhat wider margin (Table 5.10), even though Levy and Sandler's algorithm had a higher precision and recall, on average. This unlikely outcome is possibly due to the fact that the average *f*-measure is not equal to the average precision and recall scores. Incidentally, this is the reason that the *f*-measure for Levy and Sandler (2007) was not estimated from the published precision and recall scores (see Tables 5.8–5.9).

Table 5.12 shows that Turnbull et al. (2007) still top the other algorithms in boundary estimation quite impressively, with a boundary *f*-measure (0.38) nearly double that of the next-best algorithm (0.22, from Barrington 2010). Unlike the algorithms evaluated here, Turnbull et al. used supervised learning. They viewed boundary estimation as a classification problem: based on labelled RWC data, an algorithm was trained to learn which features (from an assortment of over 800) best distinguished short audio frames that contained annotated boundaries from those that did not. That none of the other algorithms come close to matching the result achieved by Turnbull et al. suggests that improving performance may require other algorithms to use supervised learning as well.

**Segmentation results for "With the Beatles" using original TUT ground truth**

| Algorithm and parameters | *f*-measure | precision | recall |
|---|---|---|---|
| Levy et al. 2007 (timbre features) | N.A. | 0.670 | 0.640 |
| Barrington et al. (MFCCs, k=6, fine) | 0.586 | 0.544 | 0.692 |
| Levy and Sandler (CQT, k=4, N=4) | **0.634** | 0.523 | 0.835 |
| Peiszer (MFCC, k=3, k-means) | 0.595 | 0.508 | 0.734 |
| The Echo Nest | 0.452 | 0.55 | 0.401 |
| Mestres | 0.367 | **0.789** | 0.264 |
| Baseline (one boundary per second) | 0.577 | 0.409 | **1.000** |

Table 5.8. Best obtained boundary *f*-measure and corresponding precision and recall using a threshold of 3 seconds, for the 14-song album "With the Beatles." Results include those published in Levy et al. (2007), although average *f*-measure was not available. The highest values are in boldface. The original version of the TUT ground truth was used.

**Segmentation results for "Sgt. Pepper's Lonely Hearts Club Band" using original TUT ground truth**

| Algorithm and parameters | *f*-measure | precision | recall |
|---|---|---|---|
| Levy et al. 2007 (timbre features) | N.A. | **0.610** | 0.720 |
| Barrington et al. (MFCCs, k=6, coarse) | 0.518 | 0.489 | 0.603 |
| Levy and Sandler (CQT, k=10, N=4) | 0.602 | 0.456 | 0.924 |
| Peiszer (MFCC, k=3, k-means) | **0.634** | 0.506 | 0.900 |
| The Echo Nest | 0.462 | 0.422 | 0.562 |
| Mestres | 0.504 | 0.4911 | 0.555 |
| Baseline (one boundary per second) | 0.439 | 0.2854 | **1.000** |

Table 5.9. Best obtained boundary *f*-measure and corresponding precision and recall using a threshold of 3 seconds, for the 13-song album "Sgt. Pepper's Lonely Hearts Club Band." Results include those published in Levy et al. (2007), although average *f*-measure was not available. The highest values are in boldface. The original version of the TUT ground truth was used.

**Segmentation results for Beatles corpus using original TUT ground truth**

| Algorithm and parameters | *f*-measure | precision | recall |
|---|---|---|---|
| Barrington et al. (MFCCs, k=6, fine) | 0.555 | 0.5473 | 0.7228 |
| Levy and Sandler (MFCCs, k=5, N=4) | 0.5811 | 0.586 | 0.8324 |
| Peiszer (MFCCs, k = 3, k-means) | **0.6165** | 0.5145 | 0.8237 |
| The Echo Nest | 0.494 | 0.517 | 0.5162 |
| Mestres | 0.428 | **0.6458** | 0.3803 |
| Paulus and Klapuri 2009 | 0.55 | 0.521 | 0.612 |
| Baseline (one boundary per second) | 0.4977 | 0.3369 | **1.000** |

Table 5.10. Best obtained boundary *f*-measure and corresponding precision and recall using a threshold of 3 seconds, for the corpus of 175 Beatles songs annotated using the original version of the TUT annotations. Results include those published in Paulus and Klapuri's (2009). The highest values are in boldface.

**Labelling results for Beatles corpus using original TUT ground truth**

| Algorithm and parameters | *f*-measure | precision | recall |
|---|---|---|---|
| Paulus and Klapuri 2009 | **0.599** | **0.729** | 0.546 |
| Barrington et al. (MFCCs, k=3, fine) | 0.5726 | 0.5329 | 0.6632 |
| Levy and Sandler (CQT, k=5, N=10) | 0.5968 | 0.5995 | 0.6272 |
| Peiszer (MFCCs, k=4, k-means) | 0.5965 | 0.6107 | 0.6228 |
| Baseline (one per second, constant labelling) | 0.5068 | 0.35 | **0.9976** |

Table 5.11. Best obtained pairwise *f*-measure and corresponding precision and recall, for the corpus of 175 Beatles songs with the original version of the TUT annotations. Results include those published in Paulus and Klapuri (2009). The highest values are in boldface.

**Segmentation results for RWC using 4-label ground truth**

| Algorithm and parameters | MT2G | MG2T | *f*-measure | precision | recall |
|---|---|---|---|---|---|
| The Echo Nest | 2.969 | 4.206 | 0.110 | 0.113 | 0.119 |
| Mestres | 5.147 | 3.375 | 0.204 | 0.227 | 0.200 |
| Barrington (MFCCs, k=3, fine) | 3.233 | 3.514 | 0.193 | 0.202 | 0.217 |
| Levy and Sandler (CQT, k=4, N=10) | 4.637 | 3.908 | 0.183 | 0.190 | 0.255 |
| Peiszer (MFCCs, k=3, DTW) | 1.179 | 6.189 | 0.177 | 0.131 | 0.323 |
| Barrington 2009 (MFCCs, k=5, fine) | **1.76** | 4.06 | - | - | - |
| Barrington 2010 (MFCCs, k=5, fine) | 2.96 | **3.21** | 0.22 | 0.22 | 0.22 |
| Barrington 2010 (chroma, k=5, fine) | 4.46 | 5.69 | 0.11 | 0.1 | 0.12 |
| Turnbull et al. 2007 | 1.82 | 4.29 | **0.38** | **0.33** | **0.46** |
| Baseline (one boundary per second) | **0.2433** | 7.135 | 0.0697 | 0.0697 | 0.0362 |

Table 5.12. Best obtained boundary *f*-measure and corresponding precision and recall (using a 0.5-second threshold), and median true-to-guess and guess-to-true values, for the RWC corpus, using the 4-label version of the RWC annotations. Results include those published in Barrington et al. (2009 and 2010) and Turnbull et al. (2007). The best values are in boldface.

**Labelling results for RWC using 4-label ground truth**

| Algorithm and parameters | *f*-measure | precision | recall | Rand |
|---|---|---|---|---|
| Barrington et al. 2009 (MFCCs, k=4, fine) | - | - | - | **0.79** |
| Barrington et al. 2010 (MFCCs, k=6, fine) | 0.62 | - | - | 0.75 |
| Barrington et al. (MFCCs, k=4, fine) | 0.6219 | 0.5978 | **0.6642** | 0.7615 |
| Levy and Sandler (CQT, k=4, N=10) | 0.6283 | 0.6392 | 0.6332 | 0.7867 |
| Peiszer (MFCCs, k=5, k-means) | **0.6291** | **0.6666** | 0.6085 | 0.7817 |
| Baseline (one boundary per second, constant labels) | 0.483 | 0.3197 | 1 | 0.6847 |

Table 5.13. Best obtained pairwise *f*-measure and corresponding precision, recall and Rand index, for the RWC corpus, using the 4-label version of the RWC annotations. Results include those published in Barrington et al. (2009 and 2010), although not all values were available from these sources. The best values are in boldface.

**Segmentation results for RWC using original ground truth**

| Algorithm and parameters | *f*-measure | precision | recall |
|---|---|---|---|
| Barrington et al. (MFCCs, k=6, coarse) | 0.6106 | 0.7013 | 0.6513 |
| Levy and Sandler (MFCCs, k=10, N=4) | 0.6606 | **0.7746** | 0.7546 |
| Peiszer (MFCCs, k=3, k-means) | **0.6799** | 0.613 | 0.8067 |
| The Echo Nest | 0.5356 | 0.6785 | 0.4572 |
| Mestres | 0.4775 | 0.6857 | 0.3834 |
| Paulus and Klapuri 2009 | 0.63 | 0.717 | 0.578 |
| Baseline (one boundary per second) | 0.5705 | 0.4057 | **1.000** |

Table 5.14. Best obtained boundary *f*-measure and corresponding precision and recall (using a 3-second threshold) for the RWC corpus, using the original version of the RWC annotations. Results include those published in Paulus and Klapuri (2009). The best values are in boldface.

**Labelling results for RWC using original ground truth**

| Algorithm and parameters | *f*-measure | precision | recall |
|---|---|---|---|
| Paulus and Klapuri 2009 | **0.637** | **0.603** | 0.721 |
| Barrington et al. (MFCCs, k=6, fine) | 0.5182 | 0.411 | **0.754** |
| Levy and Sandler (CQT, k=10, N=10) | 0.6147 | 0.5627 | 0.7078 |
| Peiszer (MFCCs, k=10, k-means) | 0.5938 | 0.5942 | 0.6171 |
| Baseline (10 random boundaries, k=10, random labelling) | 0.3788 | 0.3658 | 0.4077 |

Table 5.15. Best obtained pairwise *f*-measure and corresponding precision and recall for the RWC corpus, using the original version of the RWC annotations. Results include those published in Paulus and Klapuri (2009). The best values are in boldface.

## 5.3 Discussion

Although further analysis is certainly warranted on this large set of evaluation data, some interesting conclusions may already be drawn from it. First, it was noted that the random baseline was evaluated very favourably on the condensed versions of the ground truth in certain corpora, suggesting that the effort to generate these ground truth variations was misguided. That is, it is a misreading of the Beatles annotations to collect sections with related labels, such as "refrain" and "refrain_solo," and conflate them as being the same. Since on the other corpora most algorithms performed about equally well using any version of the ground truth, the condensed and the four-label versions of the ground truth were set aside.

Second, an analysis of the importance of setting the parameter *k* confirmed that performance improved when a value for *k* was chosen that was nearer the actual value for a particular corpus. For instance, high values of *k* led to better performance on the RWC and the small-scale Internet Archive annotations, and both collections had a high number of section types per song; the Beatles annotations generally had far fewer section types per song, and increasing *k* either had no effect or tended to hinder performance. However, the Rand index scores did not reflect this trend and seemed in all cases to favour setting a

higher value for *k*. This suggests that this metric does not weigh over-segmentation and under-segmentation errors equally, and tends to disregard over-segmentation errors.

Thirdly, with various simplifications afforded by these observations, the importance of each algorithm's input parameters was assessed. Barrington et al.'s algorithm was found to perform better on popular music using MFCCs, and better on classical and jazz music using chroma, whereas the choice of feature had little impact on the performance of Levy and Sandler's algorithm, and Peiszer's algorithm always performed best using MFCCs. The coarse and fine-grain output of Barrington's algorithm were found to be very similar, meaning that a good deal of time may be saved by ignoring or modifying that system's fine-grain analysis step. The most important input parameter for Levy and Sandler's algorithm was found to be the minimum segment length, which should be tuned to the attributes of the ground truth collection being used for optimal performance. And finally, the simplest clustering algorithm was found to work best for Peiszer's algorithm, with *k*-means clustering always outperforming the more complex DTW-based clustering.

### *5.3.1 Baseline performance*

An important issue raised by this evaluation is the surprising success of the baselines. The use of the conflated and the four-label versions of the ground truth was halted as a result of the baselines matching or surpassing the performance of the algorithms. On the other hand, the baselines still frequently performed very well on the original ground truth (see Figure A.2a-c and A.3d-e), and it is hard to explain the success of the baselines in these cases. The baselines benefit from a strong prior on the length of sections, but neither the constant nor the random labelling seems like they should lead to good results.

One possible conclusion is that certain evaluation metrics are poor gauges of the quality of an analysis. The Rand index in particular seemed to rank the baseline highly (see Figure A.3). Although it is ostensibly a balanced metric, the Rand index was observed not to punish over-segmentation errors as harshly as under-segmentation errors. Indeed, the best-performing baseline for the Rand index was always one that used up to 10 section labels and a random labelling procedure. By contrast, the pairwise *f*-measure

appeared to be robust, with the best-performing algorithms always outperforming the best baselines (see Figure A.1a-e).

It could also simply be that there remains great room for improvement for the algorithms evaluated here. In all of the results displayed in the appendix, the error bars represent standard error; the standard deviation, indicating the variance of the data, was usually far greater, obscuring the small differences in performance that this discussion has focused on so far. The great variance in performance could suggest that the algorithms are struggling to adapt to different musical situations, performing well in certain cases, and less successfully in others. Two example results are shown in Figure 5.3 and 5.4: the first is an example of a song that most algorithms performed well on, and the second is an example that most algorithms performed poorly on. Figure 5.4 shows the output of each algorithm when analyzing George Handel's *Suite in D*, a piece for string quartet that, as the annotations show, consists of a sequence of different themes, each repeated twice. None of the algorithms were able to reproduce this analysis.

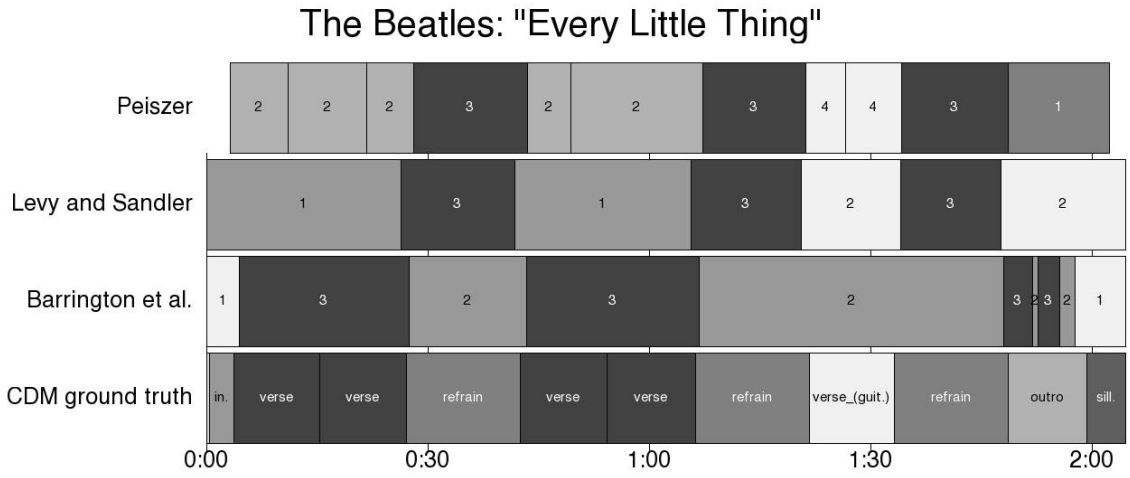



Figure 5.3. Example algorithm outputs for the Beatles song "Every Little Thing." Each algorithm output is taken from the parameter set that had the highest overall pairwise *f*-measure for the CDM annotations. For instance, the Peiszer row represents the output using $k = 4$, with MFCC features and *k*-means clustering.

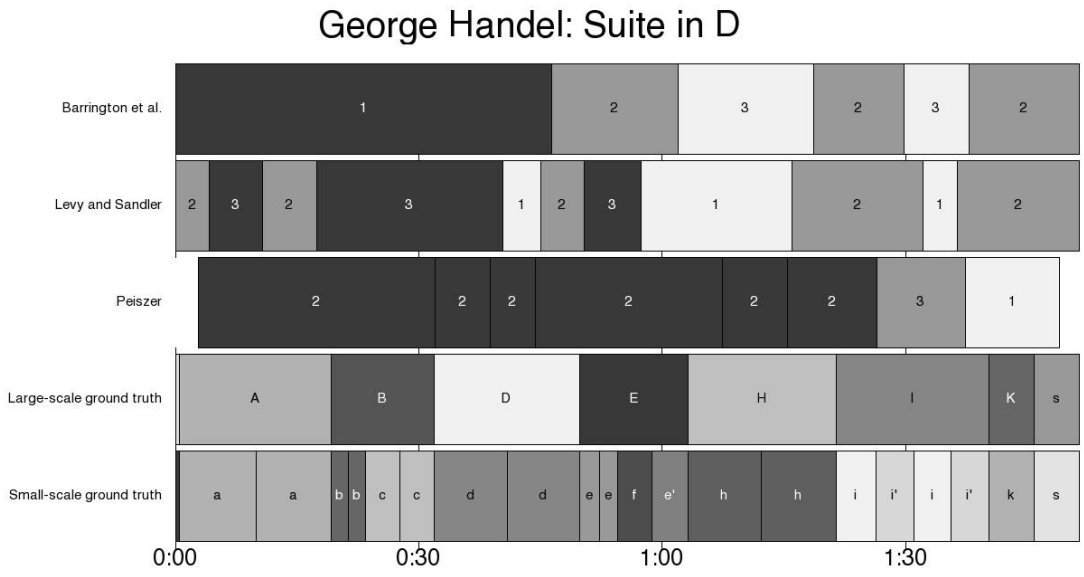



Figure 5.4. Example algorithm outputs for George Handel's *Suite in D.* Each algorithm output is taken from the parameter set that had the highest overall pairwise *f*-measure for the large-scale Internet Archive annotations.

### *5.3.2 Evaluation metrics*

The results presented in this chapter also suggest that it may be unnecessary to continue to employ such a variety of evaluation metrics. Anecdotally, it appears that pairwise *f*-measure and the purity metric *K* are strongly correlated: note the similarity between Figure A.1a and A.2a, between Figure A.1b and A.2b, and so forth; the same holds between Figure A.5a and A.5b, between Figure A.5c and A.5d, and so forth. This suggests that pairwise precision and recall, and average speaker and cluster purity, which are used to calculate the pairwise *f*-measure and purity metric *K*, may also be correlated. Investigating the covariance among the evaluation metrics merits further investigation using more rigorous statistical tools to determine which, if any, are redundant.

### *5.3.3 Difference between genres*

One of the goals of the present analysis was to establish a direct connection between these algorithms' performance on different genres of music. Encouragingly, performance appeared to be just as high on the Internet Archive corpus as it was on the Beatles and RWC corpora—indeed, it was often higher. See for instance Figures A.5a,

A.5c and A.5e, which show that the best pairwise *f*-measure for the Beatles, RWC, and large-scale Internet Archive annotations were nearly equal; note also the relative achievement on each corpus in Figures A.9a, A.10a, and A.11a, which show that Levy and Sandler in particular excelled on the large-scale Internet Archive annotations, whereas Barrington et al. and Peiszer scored comparatively higher on the Beatles corpus.

These results are reassuring in two respects: first, they suggest that the algorithms presently being designed for music structure analysis are widely applicable, despite perhaps being originally tailored to popular music. Secondly, they show that the methods that are used to compare annotated and estimated structural descriptions are not necessarily specific to a popular music context. Of course, the pieces of music in the Internet Archive corpus were explicitly chosen for their general structural coherence and simplicity, so this result may be unsurprising. Nevertheless, the result is encouraging enough to warrant expanding our test corpora further to include other challenging genres.

# 6 Conclusion

This thesis has investigated the performance of several previously published music structure analysis algorithms on a common set of three corpora, with a consistent evaluation methodology.

Chapter 2 described the principal methods that have been used to analyze audio signals and create structural descriptions algorithmically, and noted a number of practical applications that have been conceived for such algorithms.

The five analysis algorithms evaluated in this work were described in detail in Chapter 3. They included two systems—the Echo Nest audio analysis API and an algorithm described in Mestres (2009)—that estimate the location of structural boundaries, and three algorithms—Peiszer (2007), Levy and Sandler (2008) and Barrington et al. (2009)—that estimate a full structural analysis.

The evaluation framework, including the music, annotations, and evaluation metrics used here, was discussed in Chapter 4. The corpora used were mainly restricted to music that was section-based, and included two collections of popular music (the Beatles catalogue and the RWC Popular Music Database) that have been frequently studied in the past, and for which there exist freely shareable annotations. A third was created from classical and jazz music, which is freely available on the Internet Archive, and provided with two sets of annotations describing the large- and small-scale structure of these pieces. With this choice of corpora, it is hoped that future evaluations may be easily compared to the present one.

The performance metrics considered here, whose derivation and attributes were described in Section 4.3, included most of those used in previous evaluations of structure analysis algorithms.

The algorithms described in Chapter 3 were executed on the three corpora described in Chapter 4, and their performance was evaluated using the metrics described in Section 4.3. The evaluation included several naïve baseline analysis approaches drawn from previous studies. The full results of this evaluation are available online, and Chapter 5 presented and examined some of this data from a number of perspectives.

First, a view of how each algorithm's performance varied with the different versions of the ground truth (Figures A.1–A.4) showed that two proposed manipulations of the annotations appeared to be oversimplifications. This was evidenced by the fact that the naïve baseline performance was much higher with the simplified versions, while the performance of the algorithms changed comparatively little.

Second, examining how performance varied with the input parameter *k* specifying the maximum number of section types showed that, as expected, algorithms performed better when the input parameter *k* was nearer to the actual average *k* for each corpus (Figures A.5–A.8). The fact that this trend was not reflected by the Rand index led to the unexpected suggestion that the Rand index may be an unreliable evaluation metric, prone to forgiving errors of over-segmentation.

Third, the effect of choosing different input parameters was investigated for each algorithm (Figures A.9–A.11). It was observed that different algorithms had different dependencies on comparable parameters: for instance, while Levy and Sandler's algorithm performed the same whether the CQT or MFCCs were used as features, Peiszer's algorithm consistently performed better using MFCCs, and Barrington et al.'s algorithm performed better using MFCCs on popular music, but performed better using chroma on the Internet Archive corpus.

The present evaluation raises some interesting questions that warrant further analysis. Anecdotally, it appears that some of the evaluation metrics (most prominently, pairwise *f*-measure and purity measure *K*) are correlated; an investigation of how all of the evaluation metrics covary could provide a motivation to discount certain metrics in the future, or perhaps to combine them into a single, overall quality measure. Information on this subject may be welcomed by a community that has already made note of the lack of unified reporting practice (see, e.g., Lukashevich 2008).

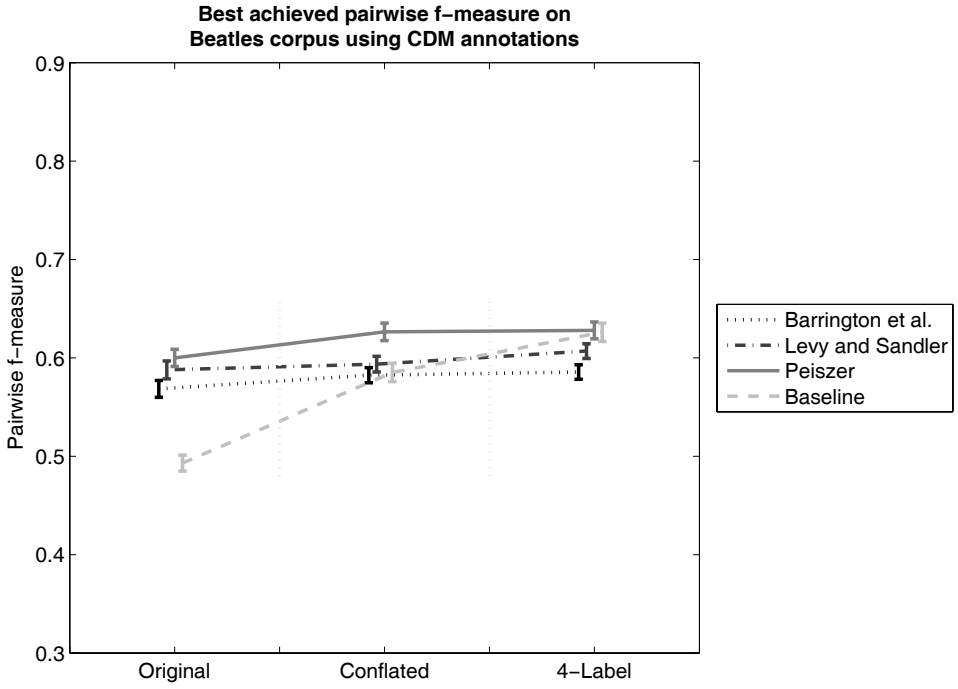
Also, while comparisons with previously published evaluations showed that the algorithms investigated here performed on par with others, one special exception is Turnbull et al.'s (2007) supervised segmentation algorithm, which performed substantially better on the RWC database (Table 5.12). Supervised learning techniques have rarely been employed in structure analysis algorithms, but Paulus and Klapuri (2009) is one among those that have done so and it too fared as well as (Tables 5.10–

5.11) or a fair amount better (Tables 5.14–5.15) than the algorithms investigated here. Given that the state of the art in structure analysis is still sometimes outperformed by a random baseline, it is tempting to suggest that exploiting supervised methods may be necessary for the current generation of algorithms to realize substantial improvements.

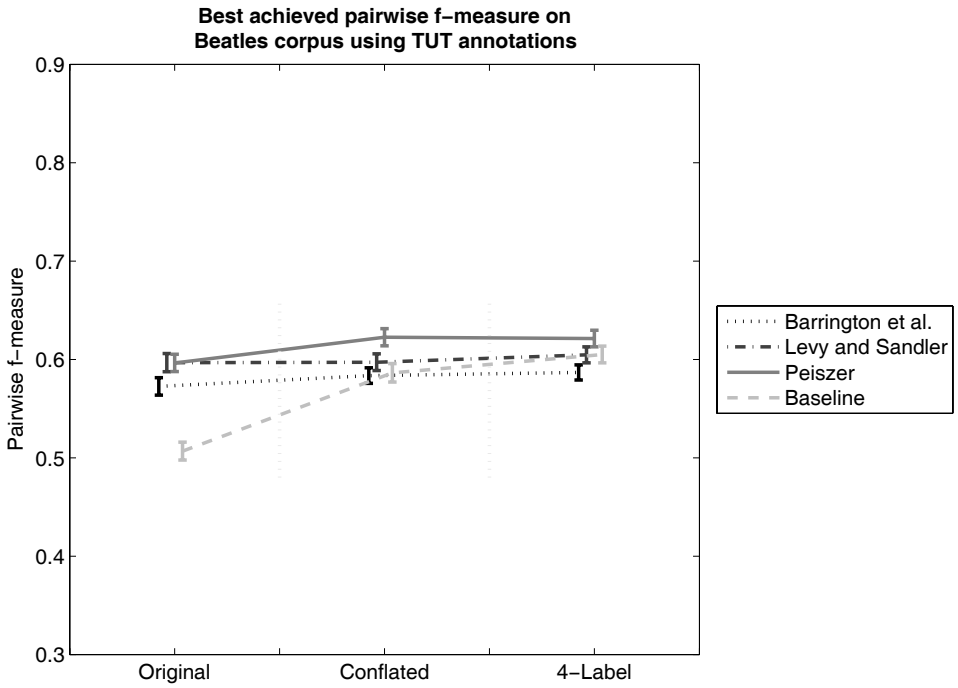
# Appendix: Plots of Results

Figure A.1, a-e. Comparison of the best average **pairwise *f*-measure** earned by any variation of each algorithm, as a function of ground truth version. Algorithms include Barrington et al., Levy and Sandler, Peiszer, and the baselines. The results are presented for each corpus of annotations. Error bars represent standard error.

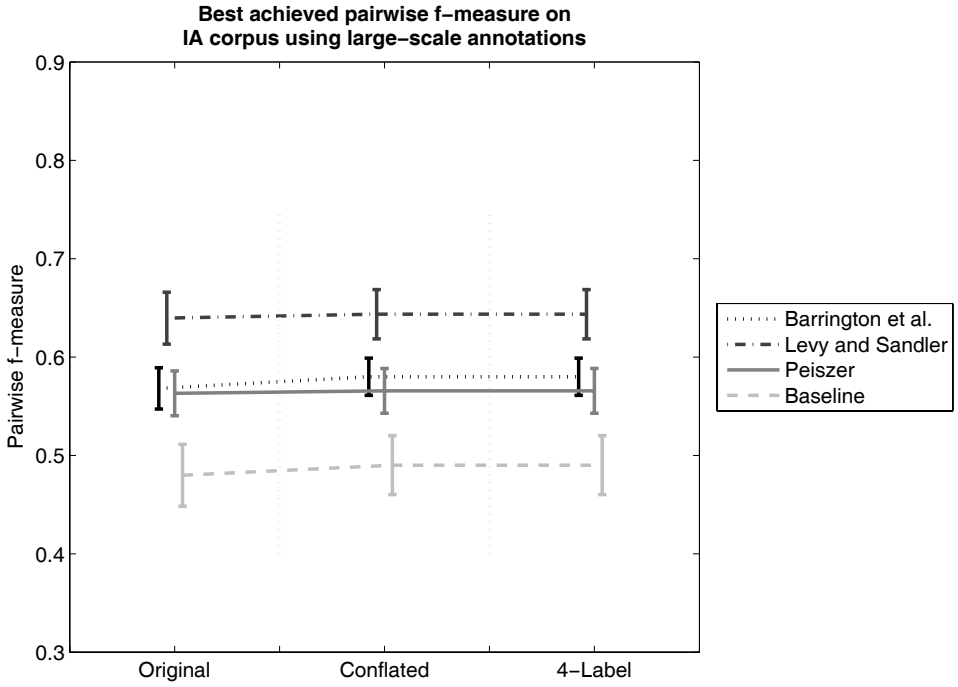
(a)



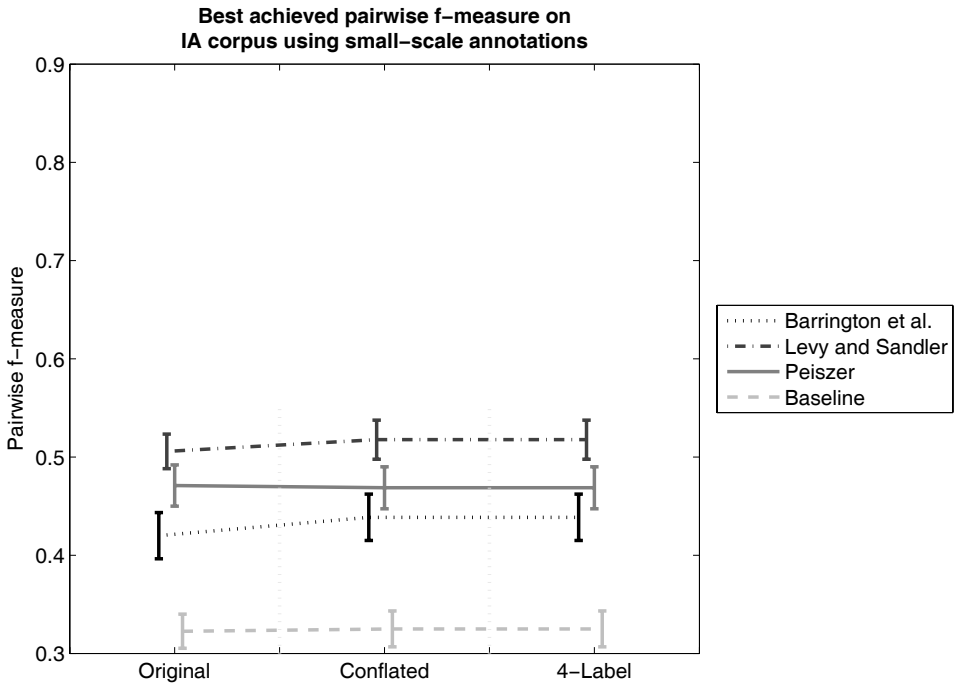
(b)


Best achieved pairwise f−measure on
Beatles corpus using TUT annotations
Pairwise f−measure
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(c)


Best achieved pairwise f−measure on
IA corpus using large−scale annotations
Pairwise f−measure
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(d)


Best achieved pairwise f−measure on
IA corpus using small−scale annotations
Pairwise f−measure
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(e)

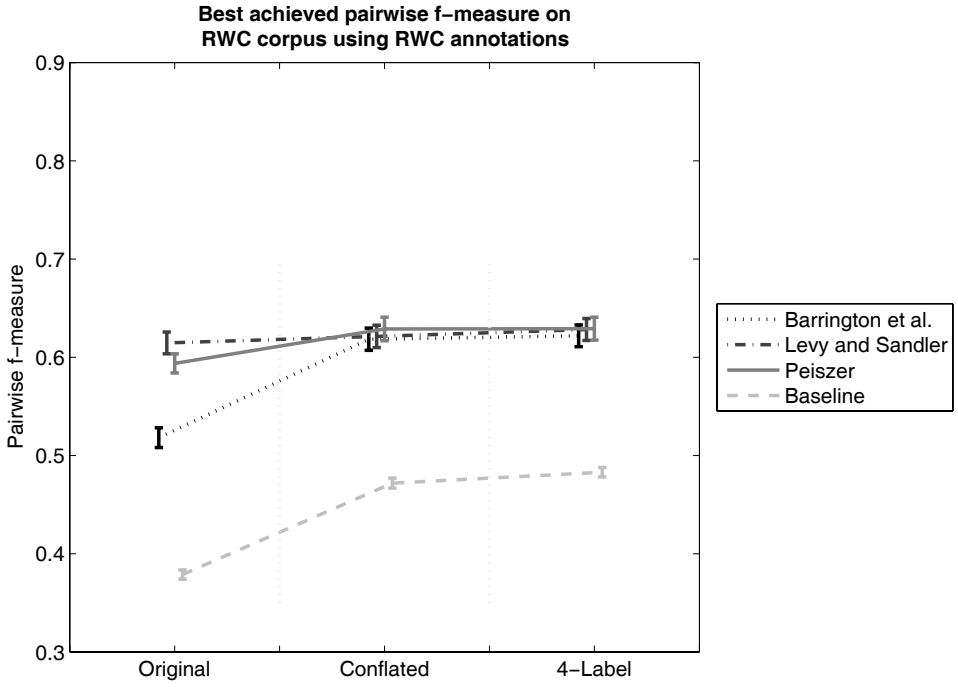
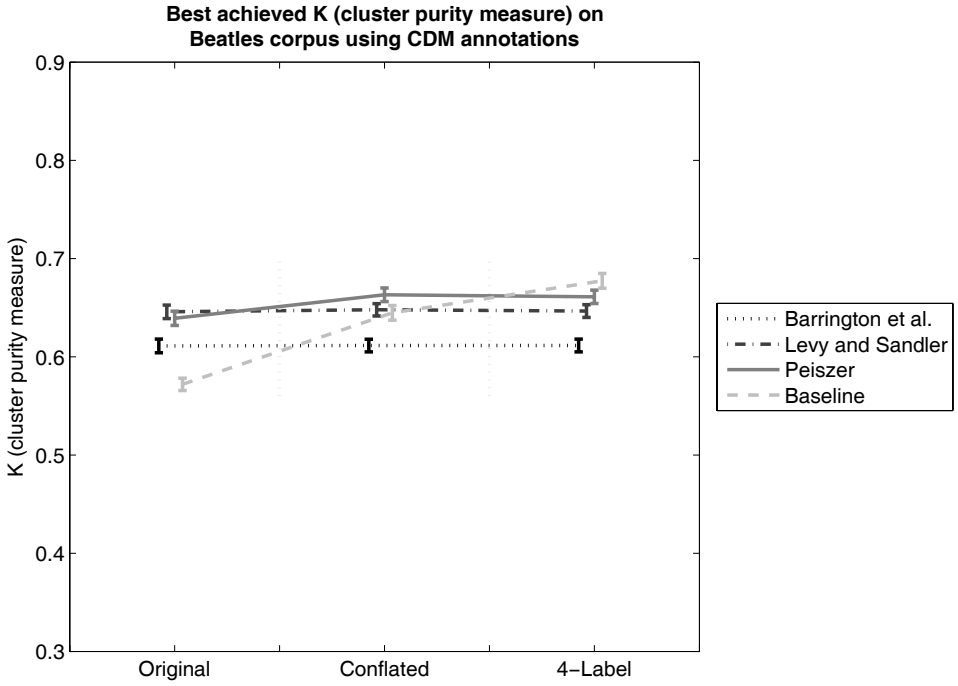
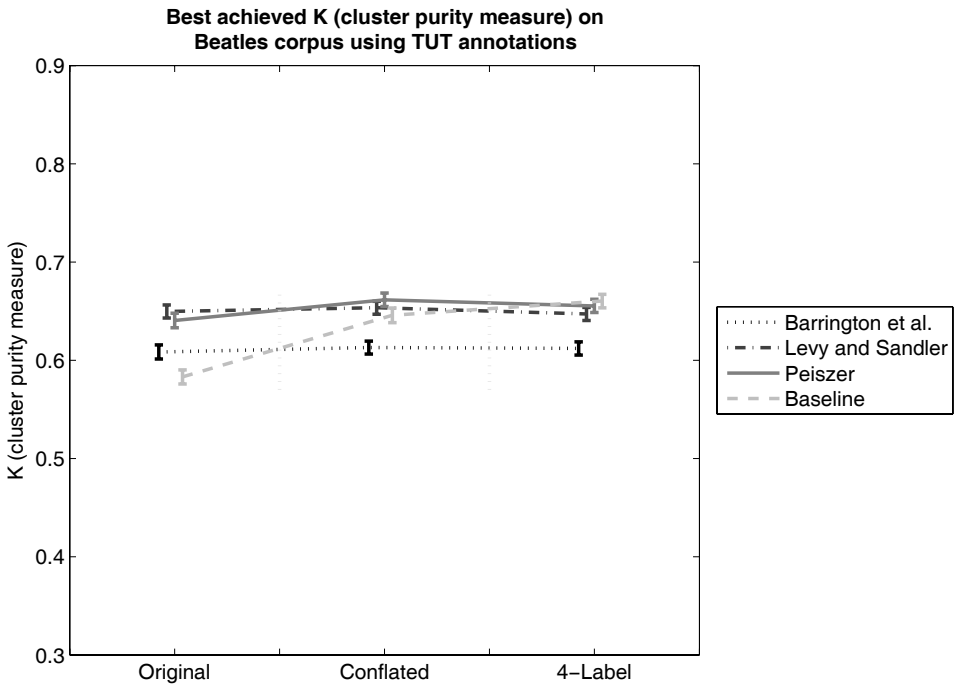

Best achieved pairwise f−measure on
RWC corpus using RWC annotations
Pairwise f−measure
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

Figure A.2, a-e. Comparison of the best average **cluster purity measure *K*** earned by any variation of each algorithm, as a function of ground truth version. Algorithms include Barrington et al., Levy and Sandler, Peiszer, and the baselines. The results are presented for each corpus of annotations. Error bars represent standard error.

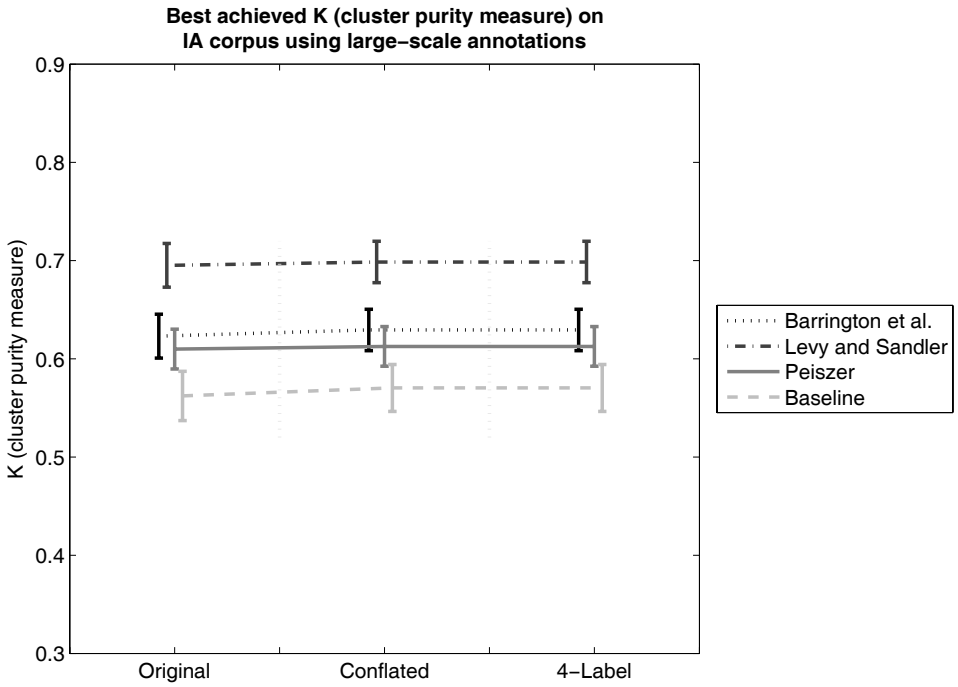
(a)



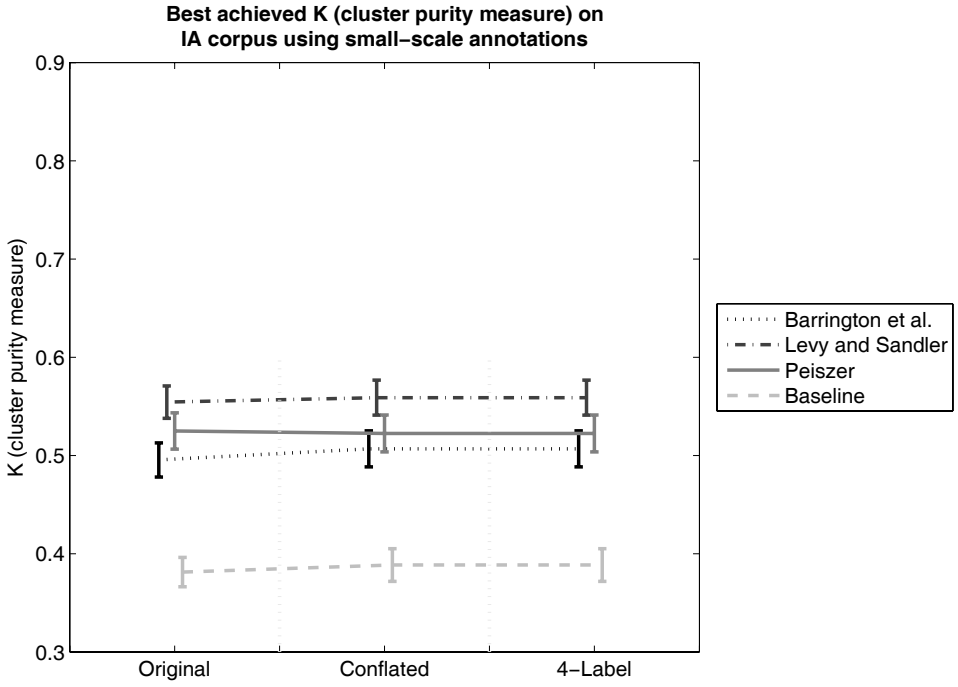

(b)



(c)


Best achieved K (cluster purity measure) on
IA corpus using large−scale annotations
K (cluster purity measure)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(d)


Best achieved K (cluster purity measure) on
IA corpus using small−scale annotations
K (cluster purity measure)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(e)

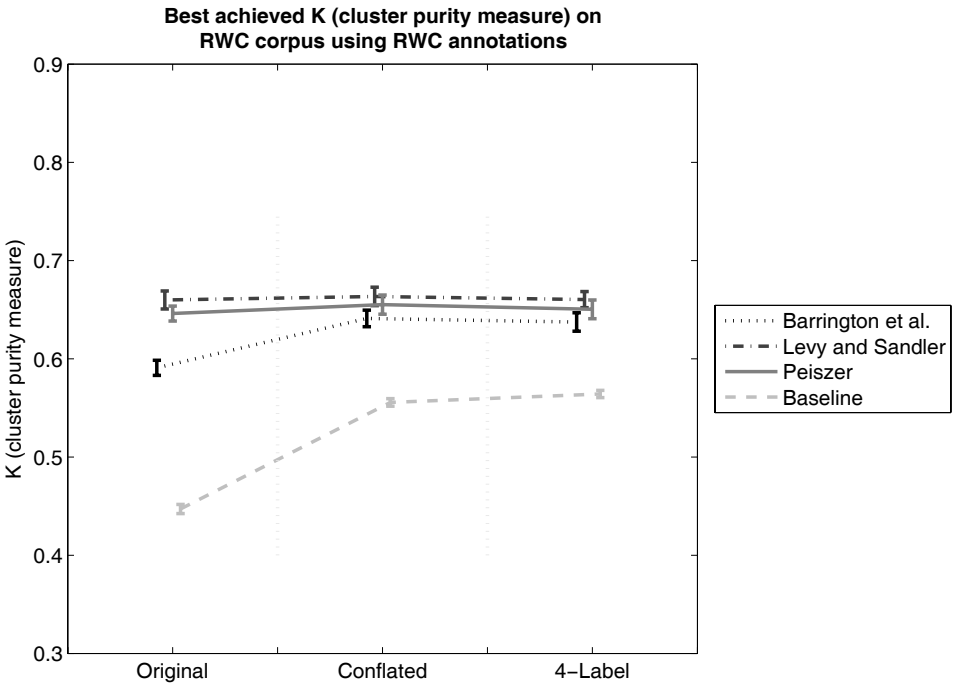
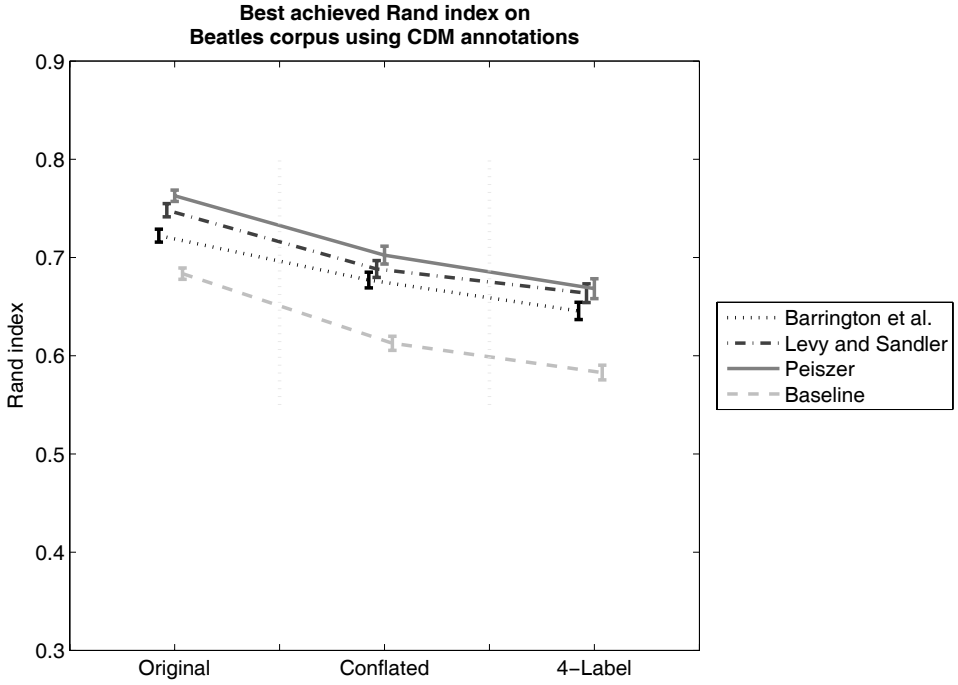
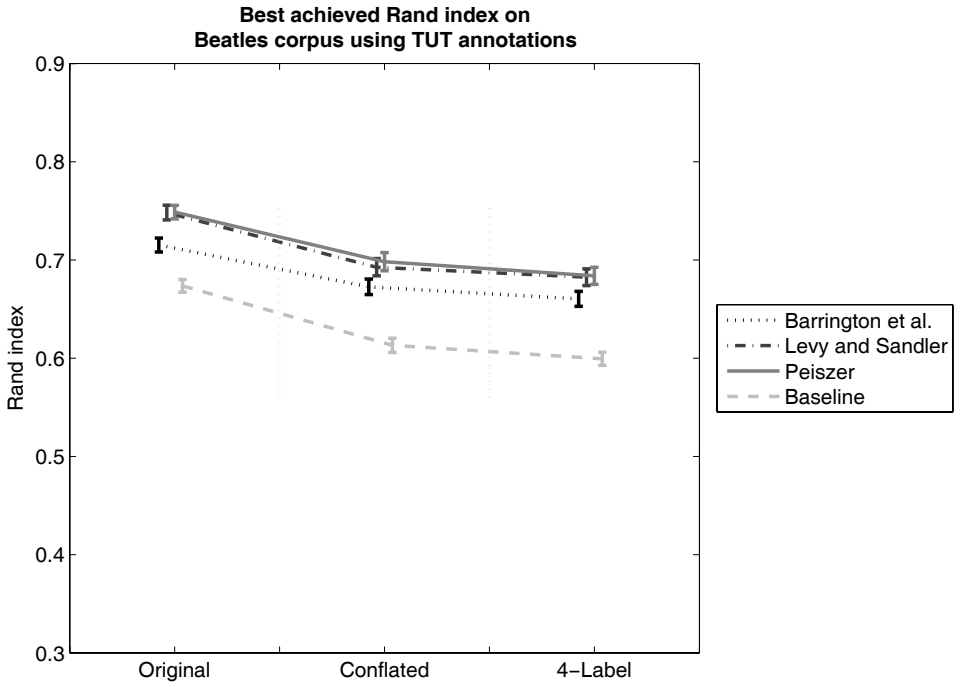

Best achieved K (cluster purity measure) on
RWC corpus using RWC annotations
K (cluster purity measure)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

Figure A.3, a-e. Comparison of the best average **Rand index** earned by any variation of each algorithm, as a function of ground truth version. Algorithms include Barrington et al., Levy and Sandler, Peiszer, and the baselines. The results are presented for each corpus of annotations. Error bars represent standard error.

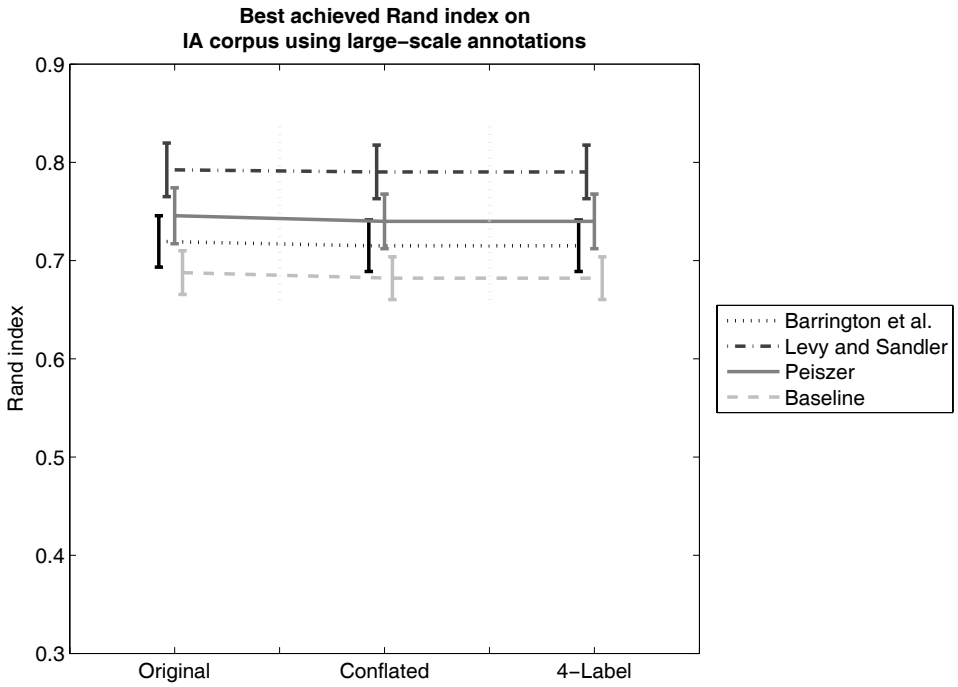
(a)



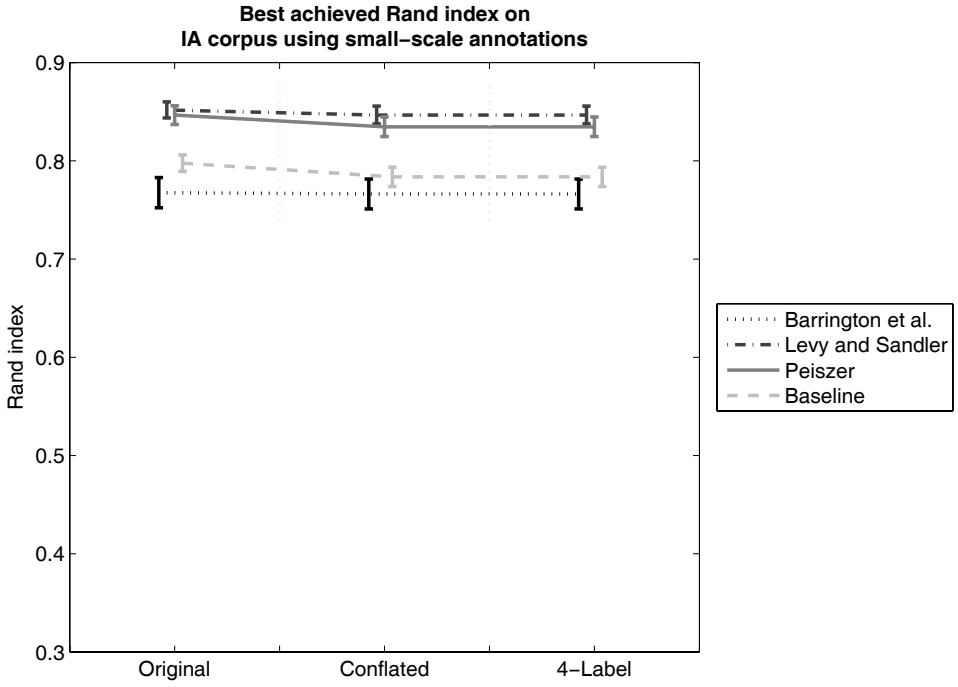

(b)



(c)


Best achieved Rand index on
IA corpus using large−scale annotations
Rand index
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(d)


Best achieved Rand index on
IA corpus using small−scale annotations
Rand index
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(e)

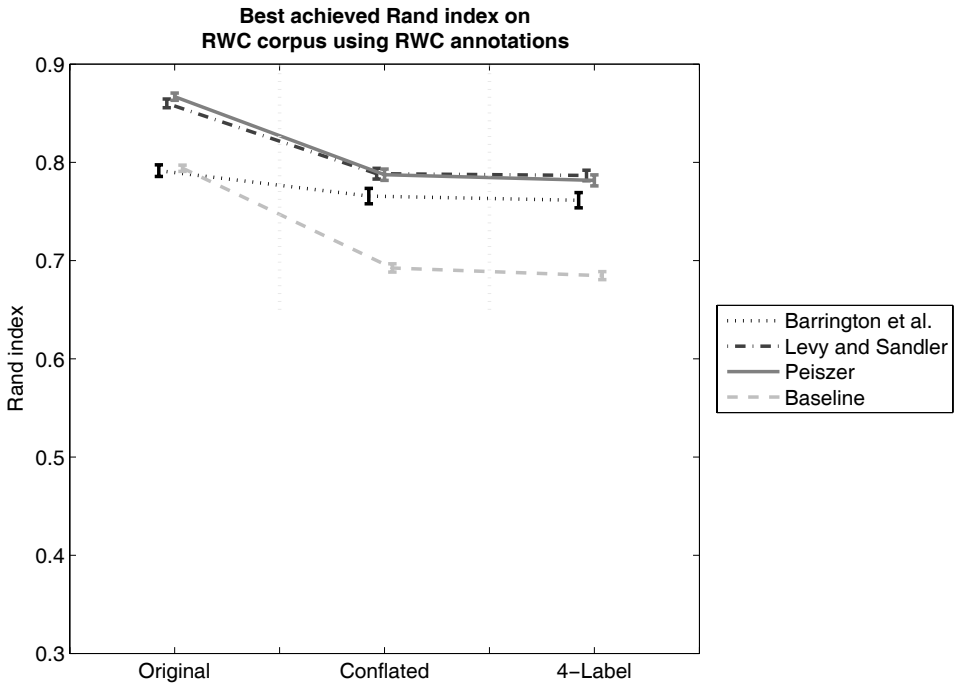
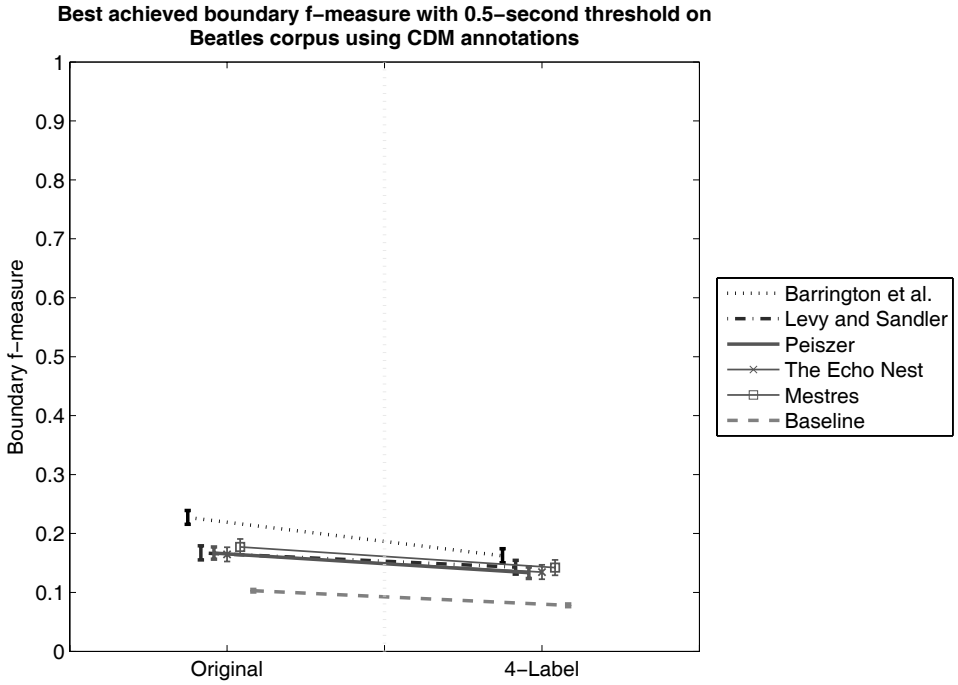
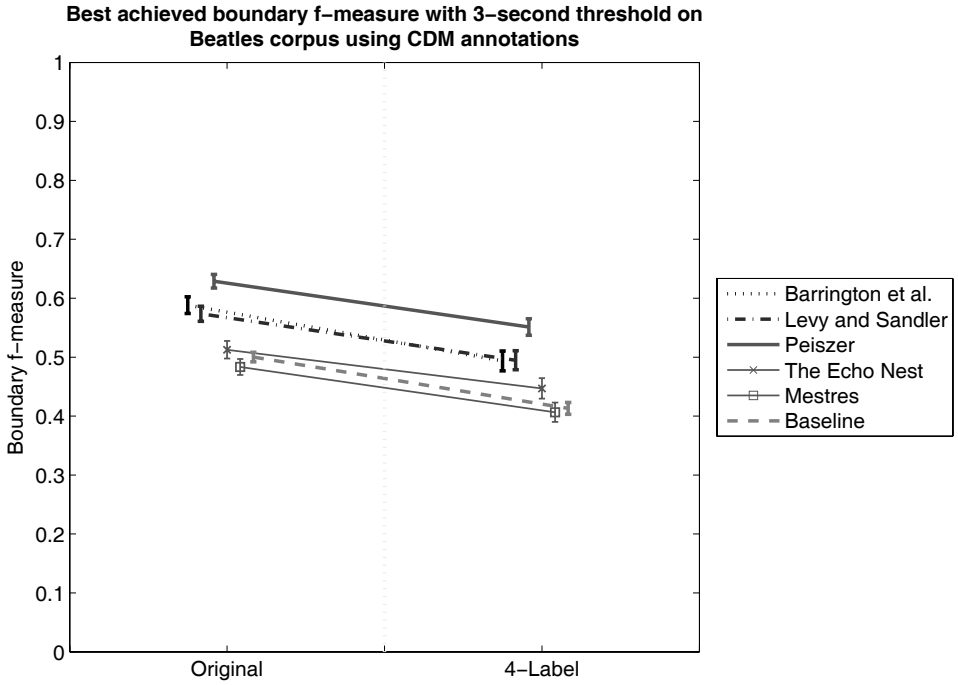

Best achieved Rand index on
RWC corpus using RWC annotations
Rand index
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Original
Conflated
4−Label
Barrington et al.
Levy and Sandler
Peiszer
Baseline

Figure A.4, a-j. Comparison of the best average **boundary *f*-measure** earned by any variation of each algorithm, as a function of ground truth version. (Note that the boundaries in the "original" and "conflated" versions of the ground truth are identical.) The results are presented for two levels of tolerance, **0.5 and 3 seconds**. Algorithms include Barrington et al., Levy and Sandler, Peiszer, The Echo Nest, Mestres, and the baselines. The results are presented for each corpus of annotations. Error bars represent standard error.

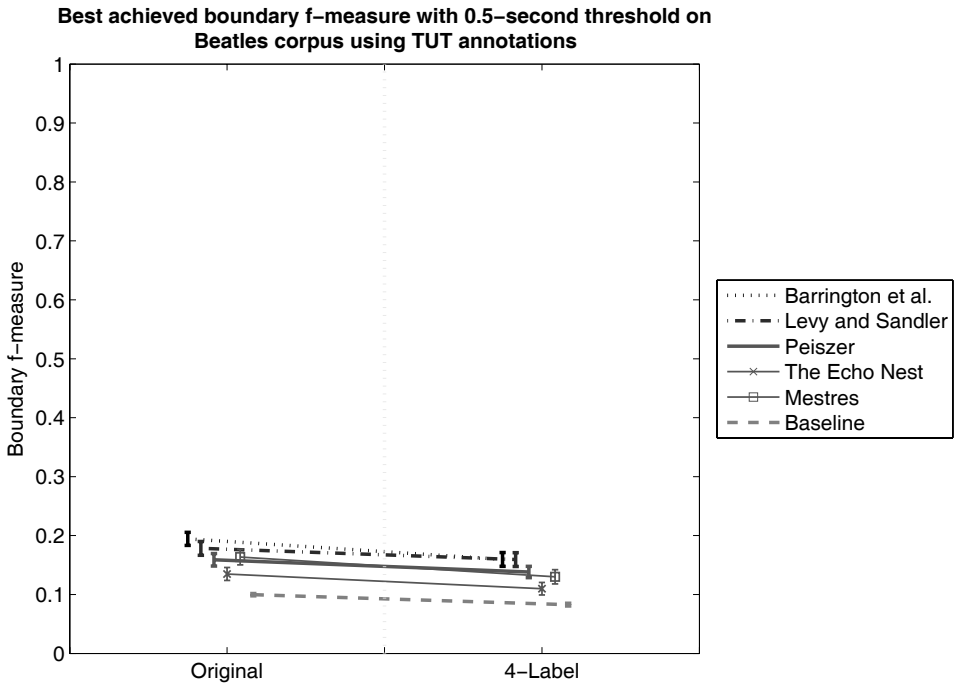
(a)



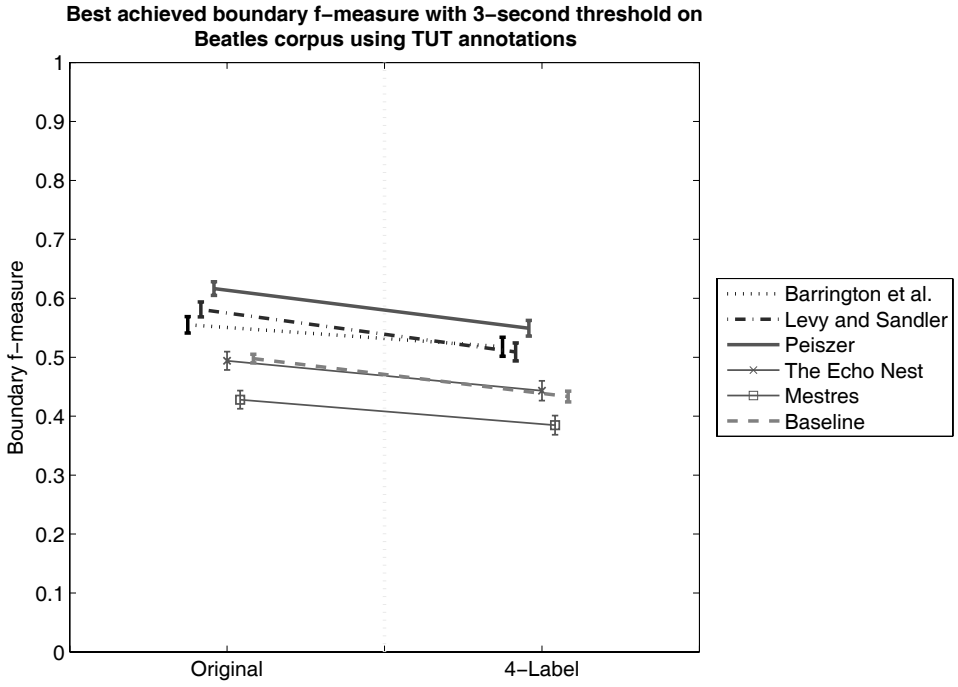

(b)



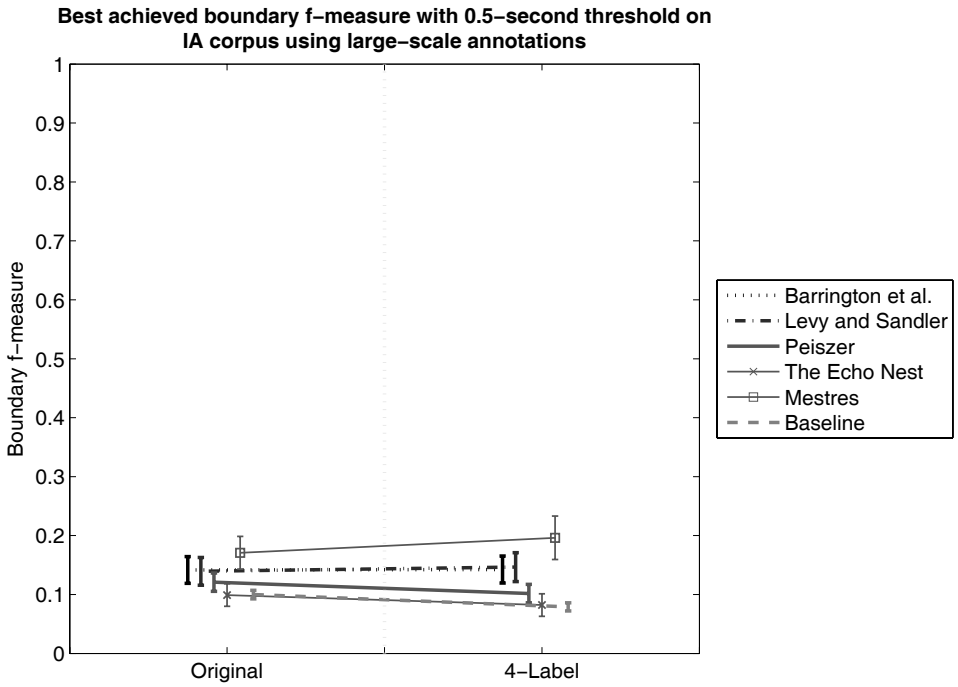
(c)


Best achieved boundary f−measure with 0.5−second threshold on Beatles corpus using TUT annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

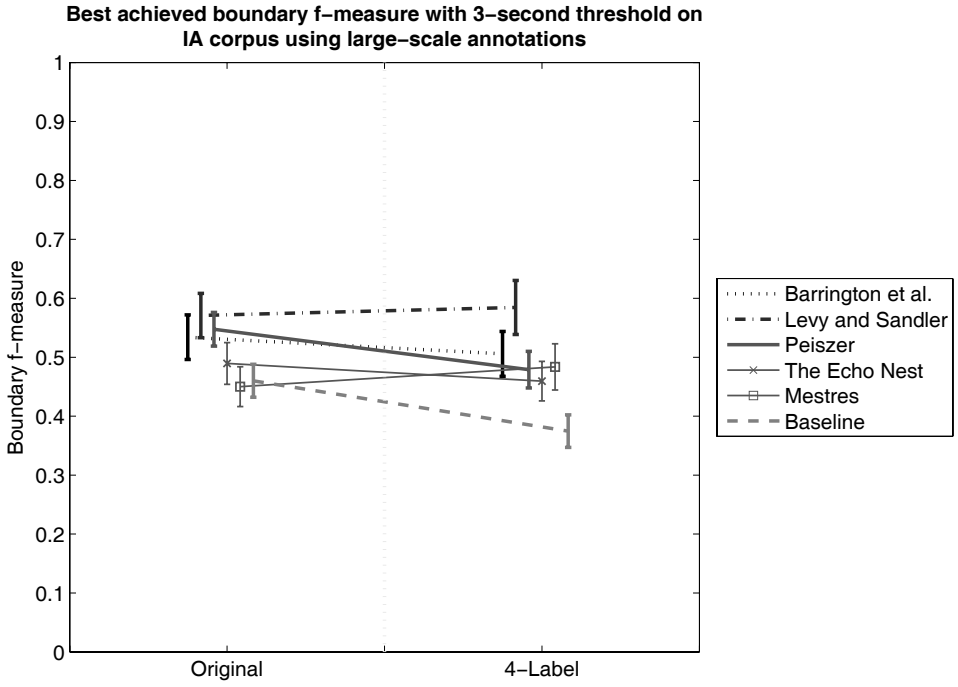
(d)


Best achieved boundary f−measure with 3−second threshold on Beatles corpus using TUT annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

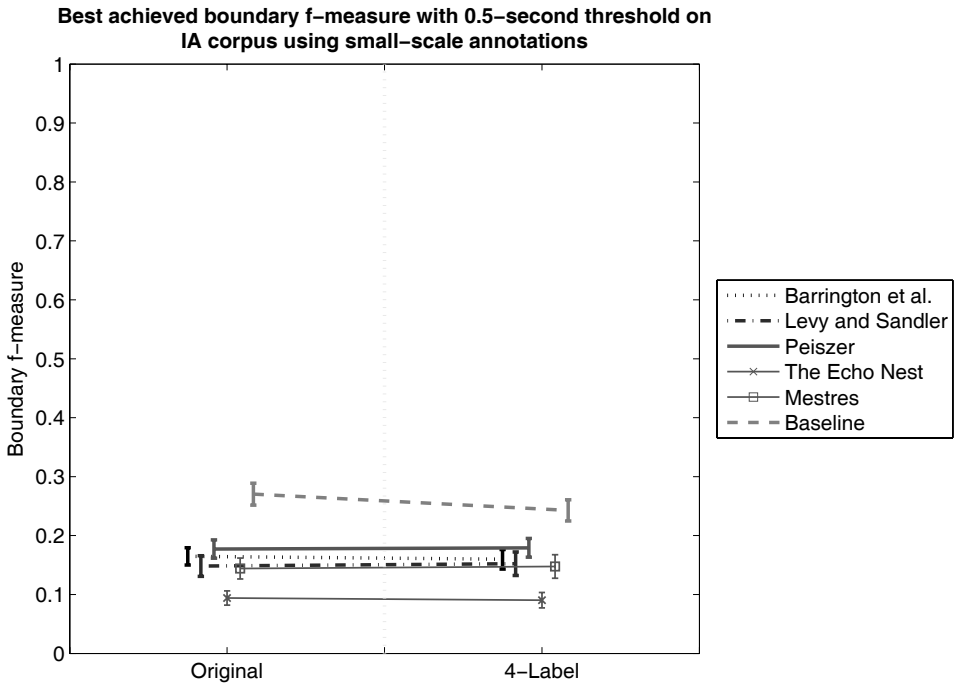
(e)


Best achieved boundary f−measure with 0.5−second threshold on IA corpus using large−scale annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

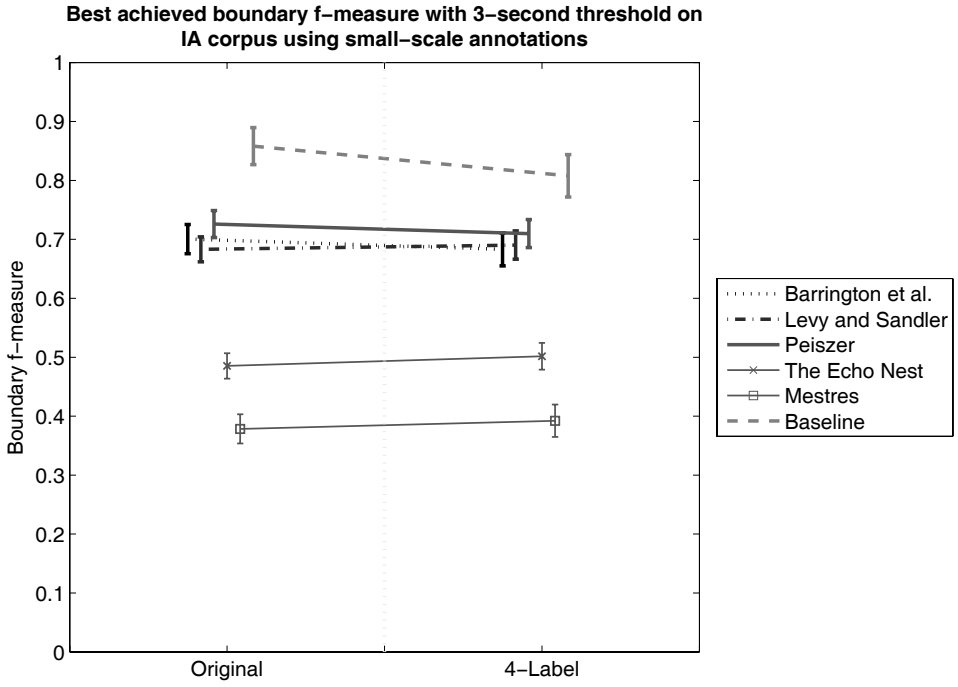
(f)


Best achieved boundary f−measure with 3−second threshold on IA corpus using large−scale annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

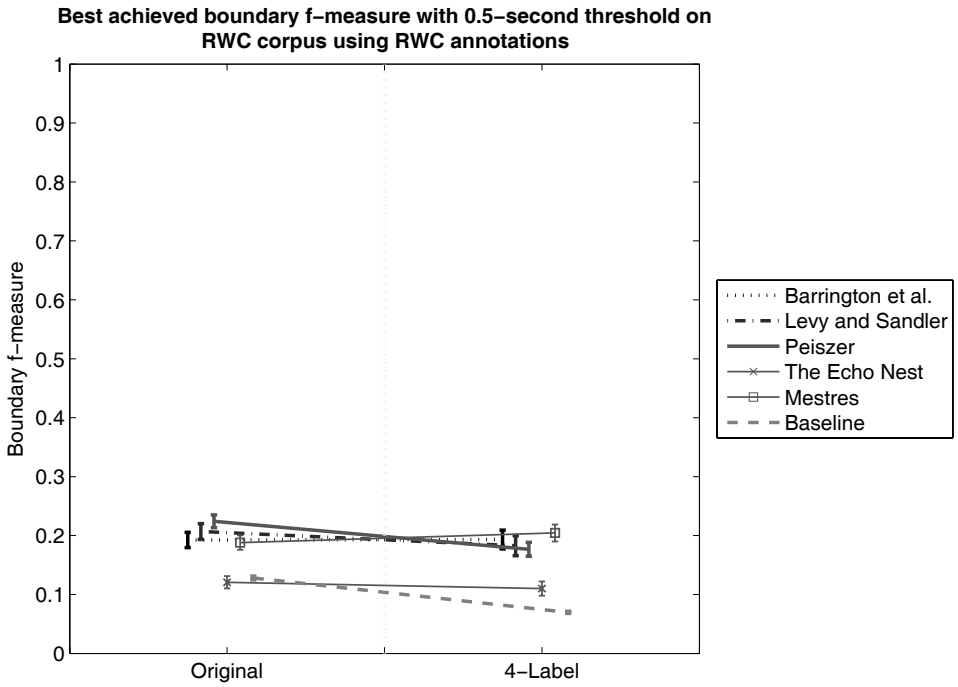
(g)


Best achieved boundary f−measure with 0.5−second threshold on IA corpus using small−scale annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

(h)


Best achieved boundary f−measure with 3−second threshold on IA corpus using small−scale annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

(i)


Best achieved boundary f−measure with 0.5−second threshold on RWC corpus using RWC annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

(j)

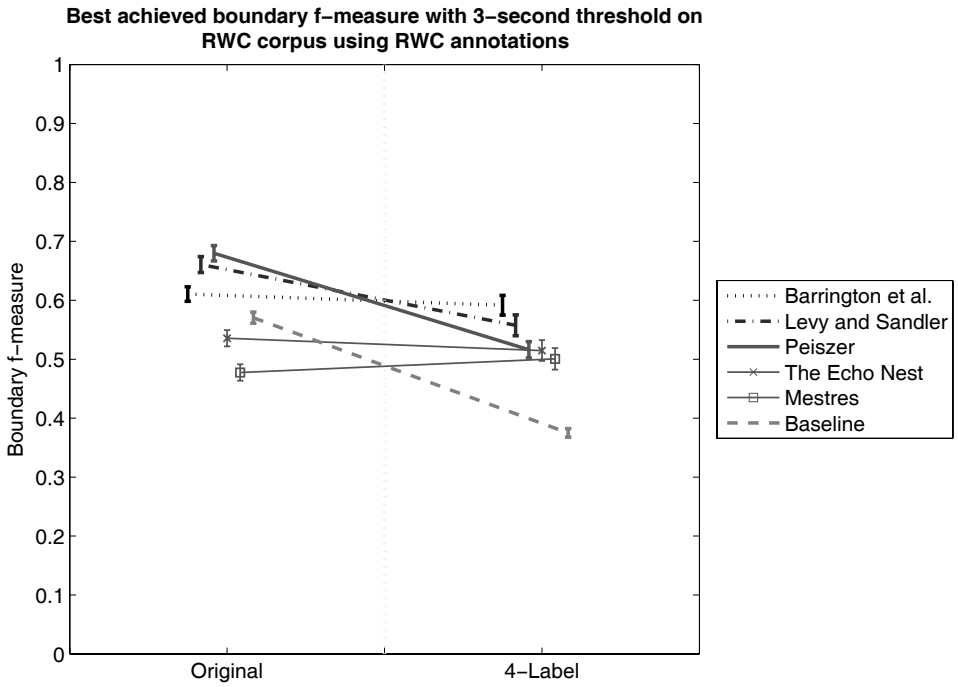
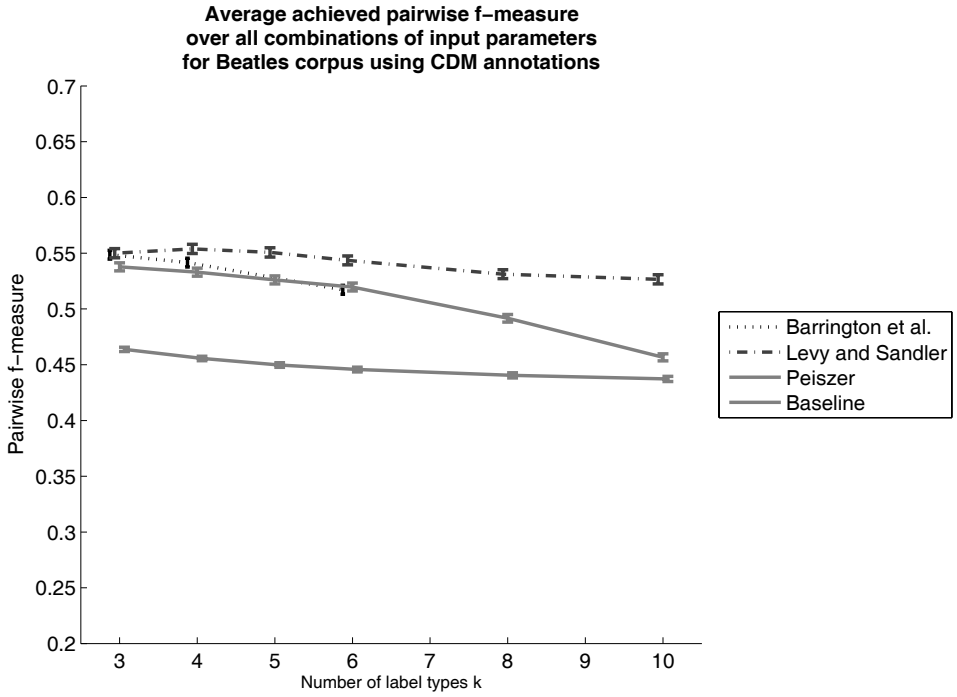
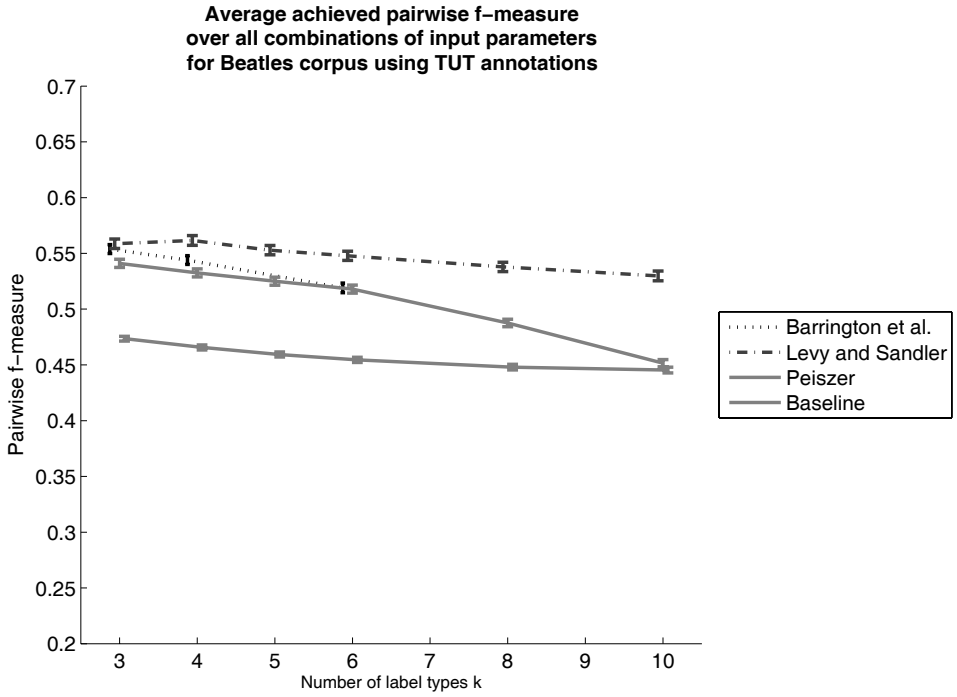

Best achieved boundary f−measure with 3−second threshold on RWC corpus using RWC annotations
Boundary f−measure
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Original
4−Label
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

Figure A.5, a-e. Comparison of the average **pairwise *f*-measure** achieved by all input parameter combinations for each algorithm as a function of the specified number of cluster types *k*. Algorithms include Barrington et al., Levy and Sandler, Peiszer, and the baselines. The results are presented for each corpus using the original versions of the annotations. Error bars represent standard error.

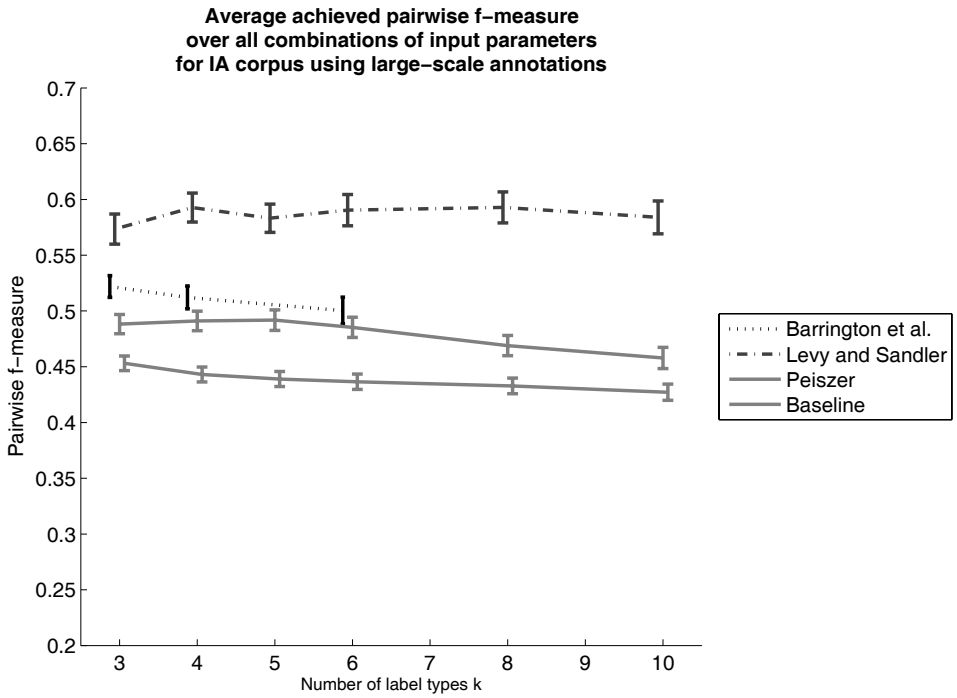
(a)



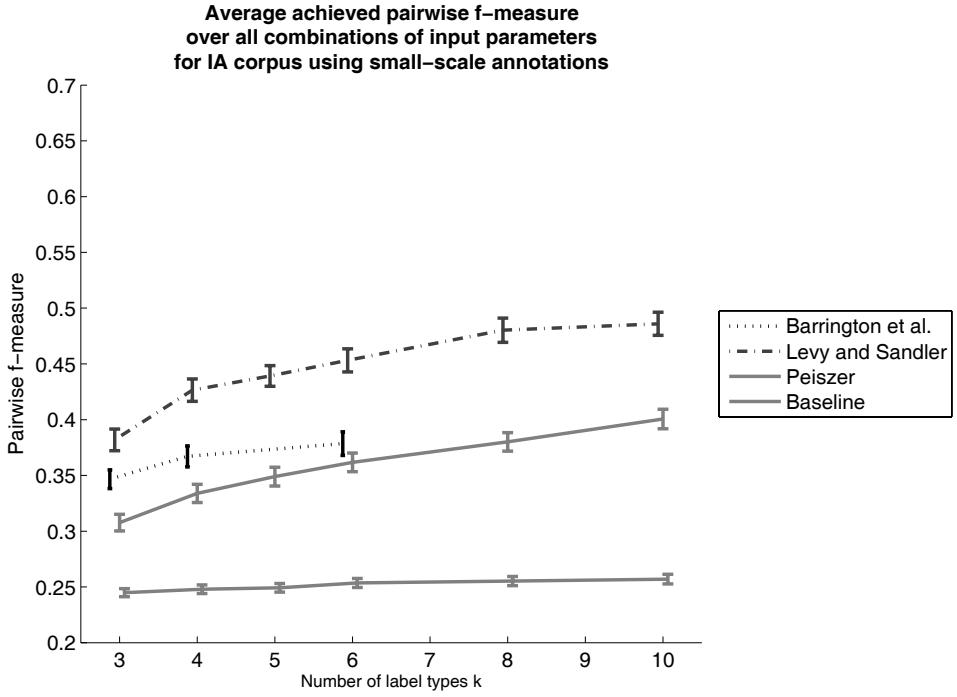

(b)



(c)


Average achieved pairwise f−measure
over all combinations of input parameters
for IA corpus using large−scale annotations
Pairwise f−measure
Number of label types k
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(d)


Average achieved pairwise f−measure
over all combinations of input parameters
for IA corpus using small−scale annotations
Pairwise f−measure
Number of label types k
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(e)

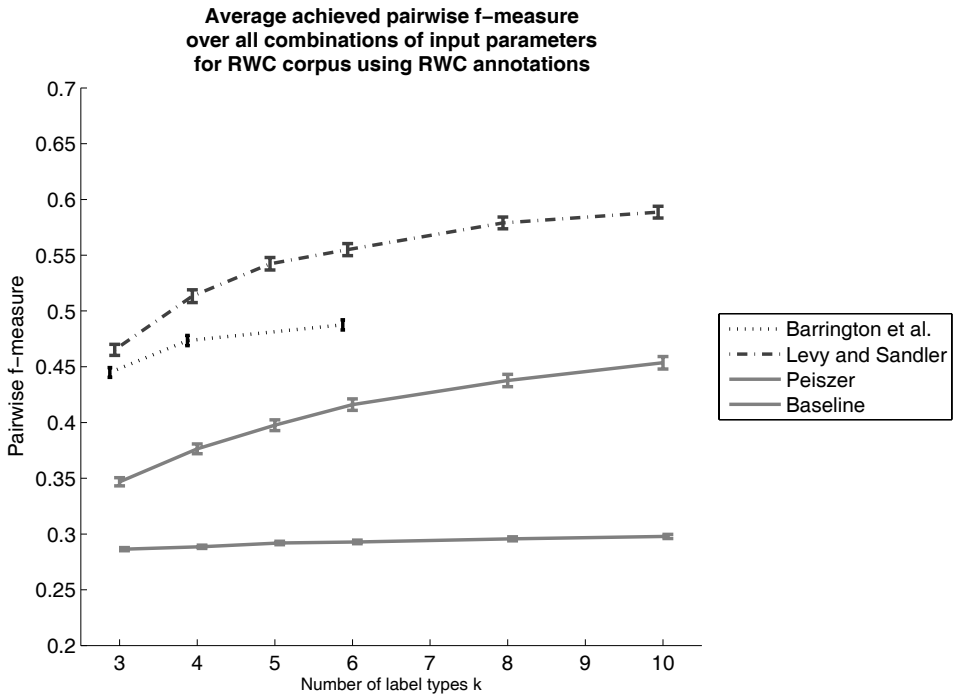
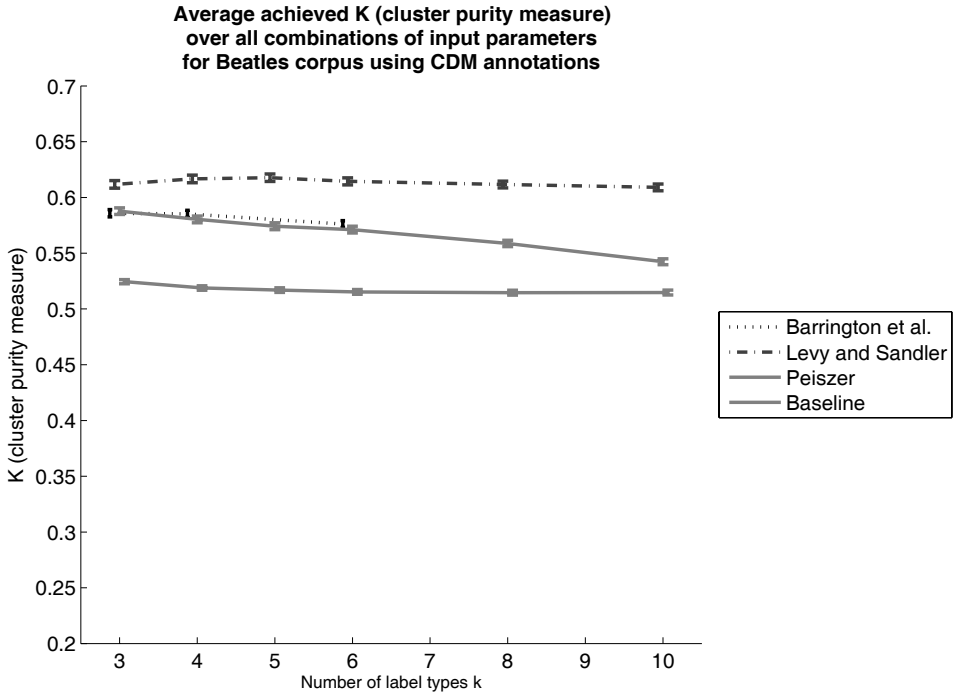
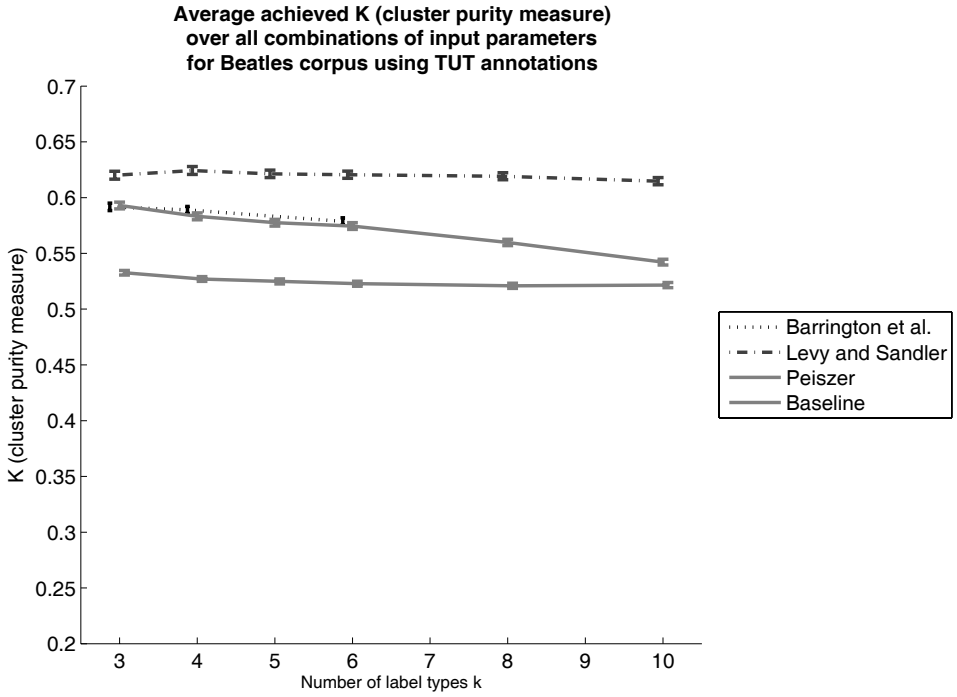

Average achieved pairwise f−measure
over all combinations of input parameters
for RWC corpus using RWC annotations
Pairwise f−measure
Number of label types k
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
Baseline

Figure A.6, a-e. Comparison of the average **cluster purity measure *K*** achieved by all input parameter combinations for each algorithm as a function of the specified number of cluster types *k*. Algorithms include Barrington et al., Levy and Sandler, Peiszer, and the baselines. The results are presented for each corpus using the original versions of the annotations. Error bars represent standard error.

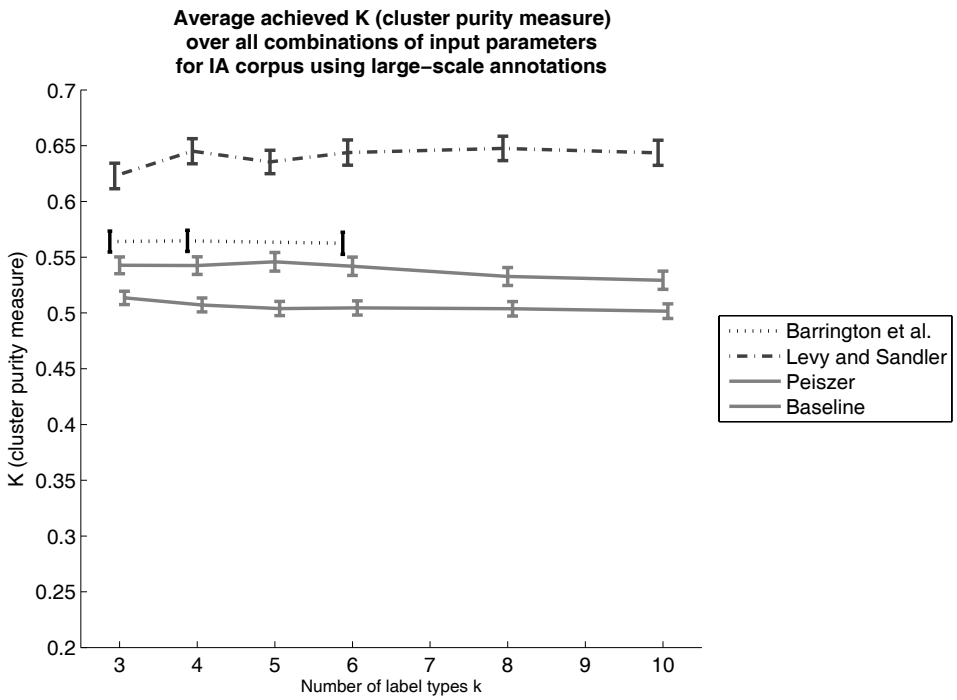
(a)



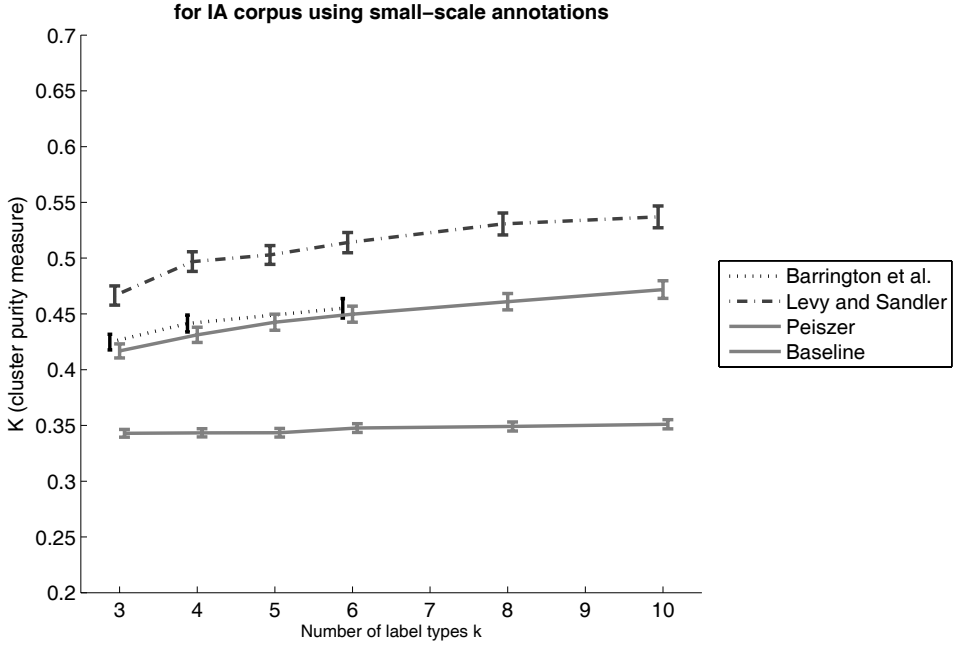

(b)



(c)


Average achieved K (cluster purity measure)
over all combinations of input parameters
for IA corpus using large−scale annotations
K (cluster purity measure)
Number of label types k
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(d)


Average achieved K (cluster purity measure)
over all combinations of input parameters
for IA corpus using small−scale annotations
K (cluster purity measure)
Number of label types k
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(e)

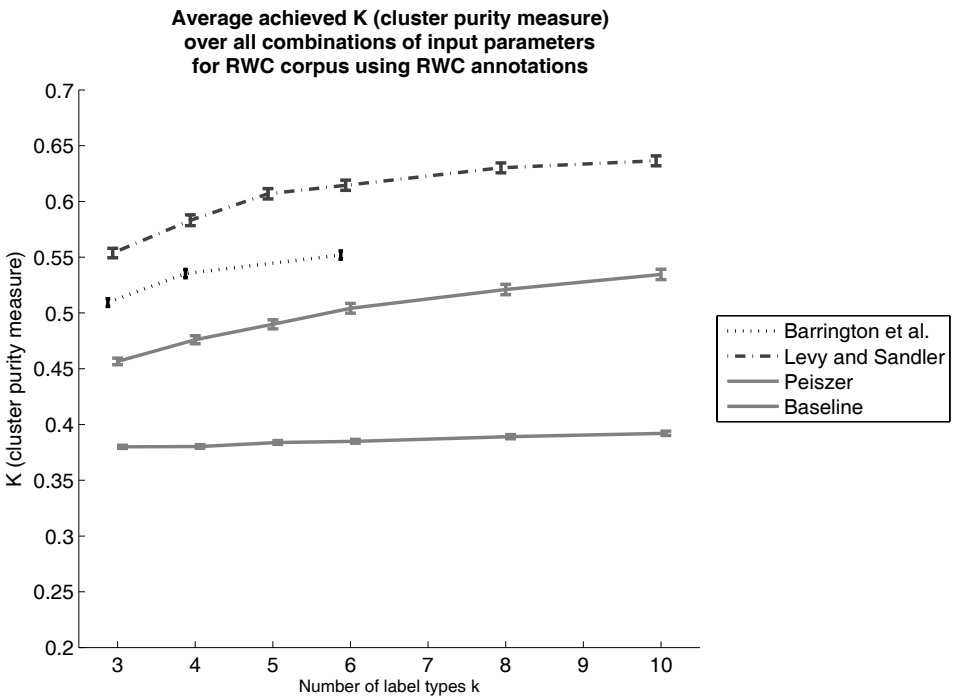
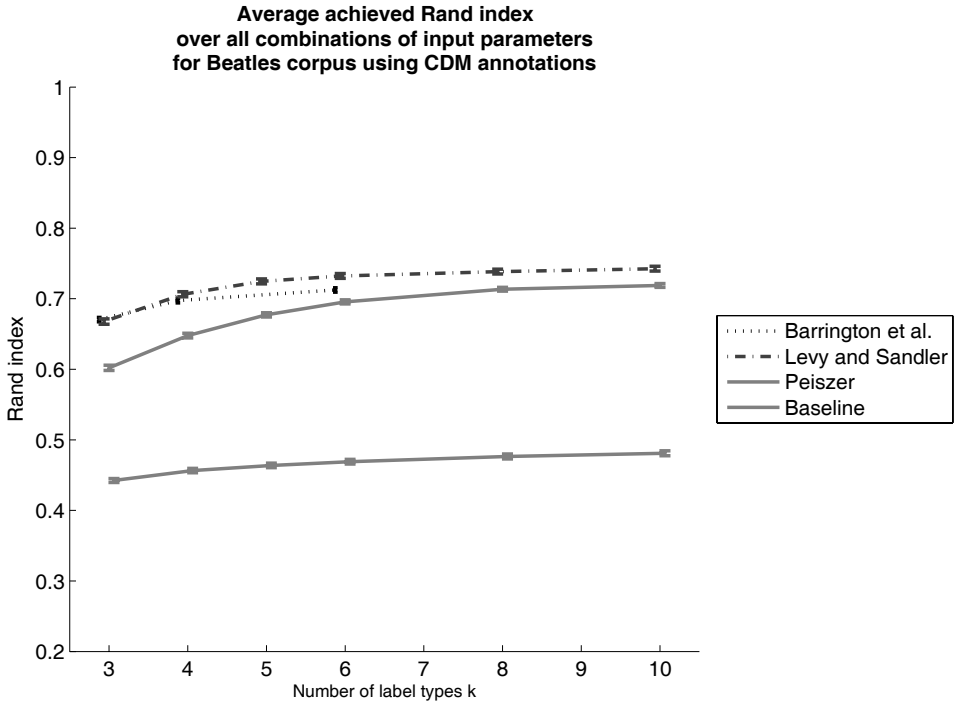
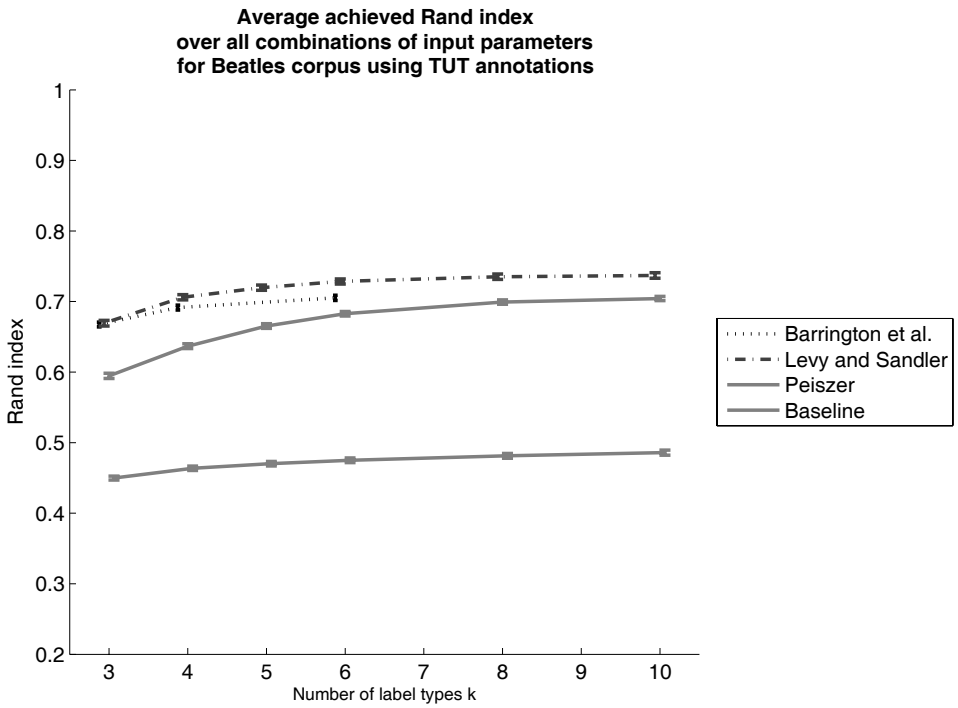

Average achieved K (cluster purity measure)
over all combinations of input parameters
for RWC corpus using RWC annotations
K (cluster purity measure)
Number of label types k
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
Baseline

Figure A.7, a-e. Comparison of the average **Rand index** achieved by all input parameter combinations for each algorithm as a function of the specified number of cluster types *k*. Algorithms include Barrington et al., Levy and Sandler, Peiszer, and the baselines. The results are presented for each corpus using the original versions of the annotations. Error bars represent standard error.

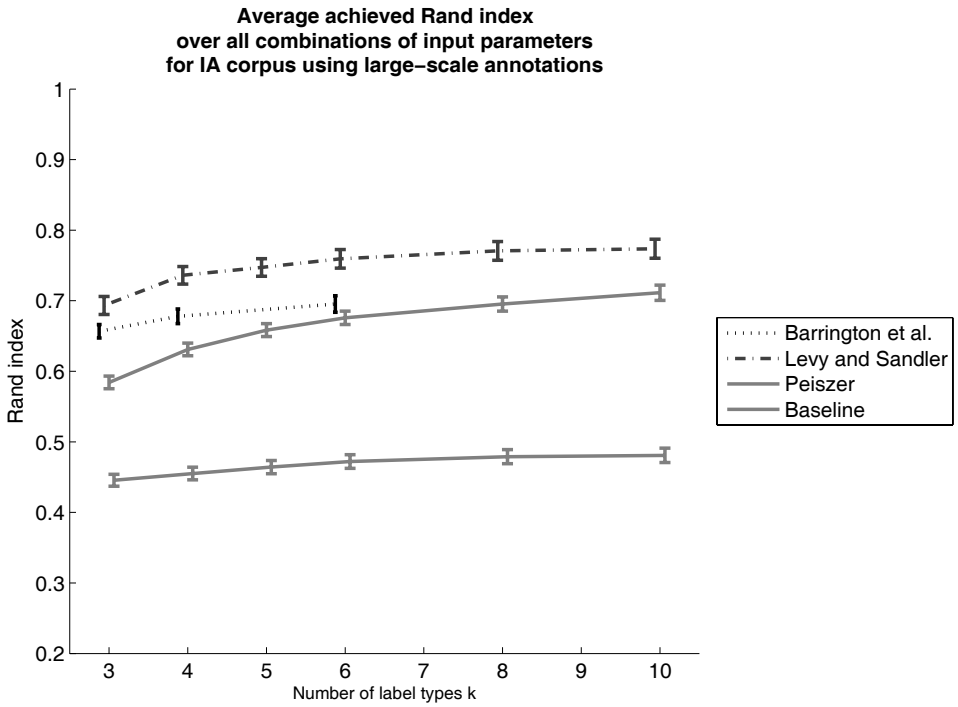
(a)



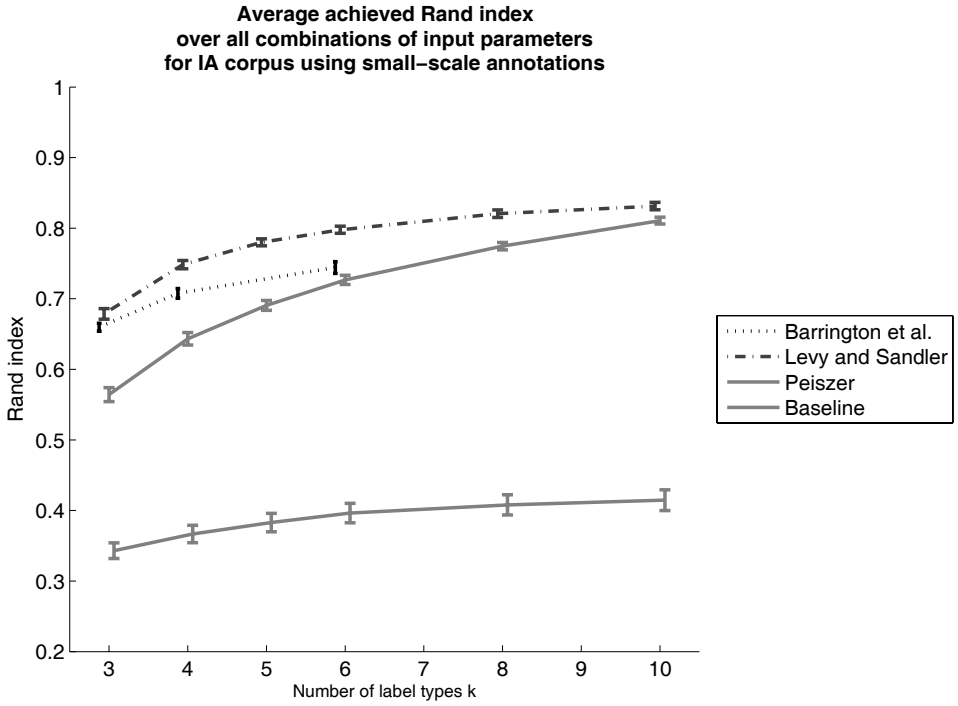

(b)



(c)


Average achieved Rand index
over all combinations of input parameters
for IA corpus using large−scale annotations
Rand index
Number of label types k
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(d)


Average achieved Rand index
over all combinations of input parameters
for IA corpus using small−scale annotations
Rand index
Number of label types k
Barrington et al.
Levy and Sandler
Peiszer
Baseline

(e)

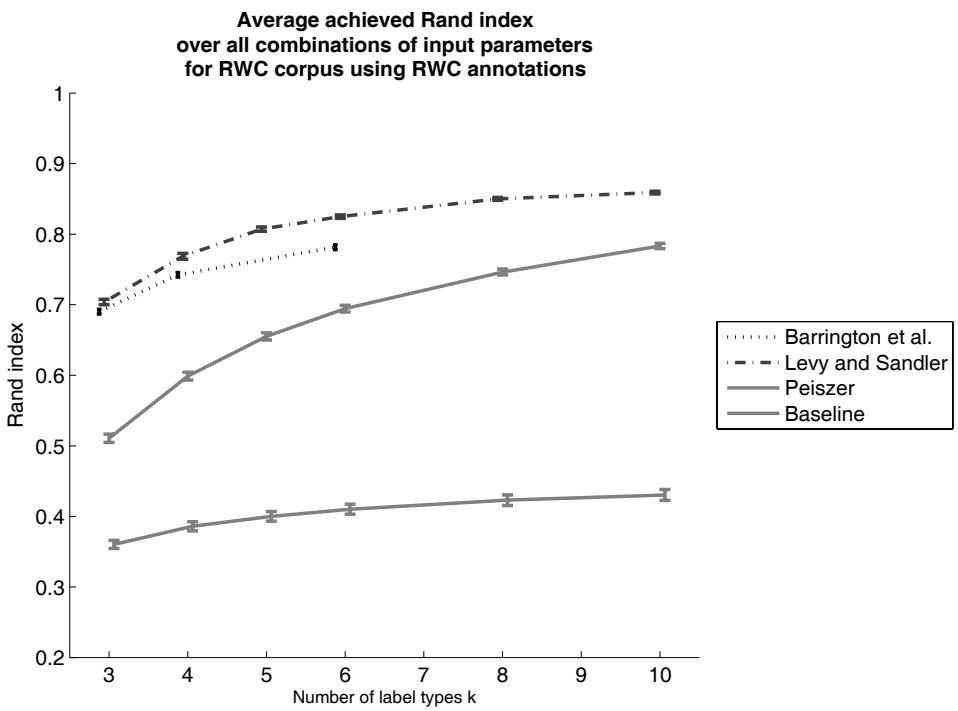
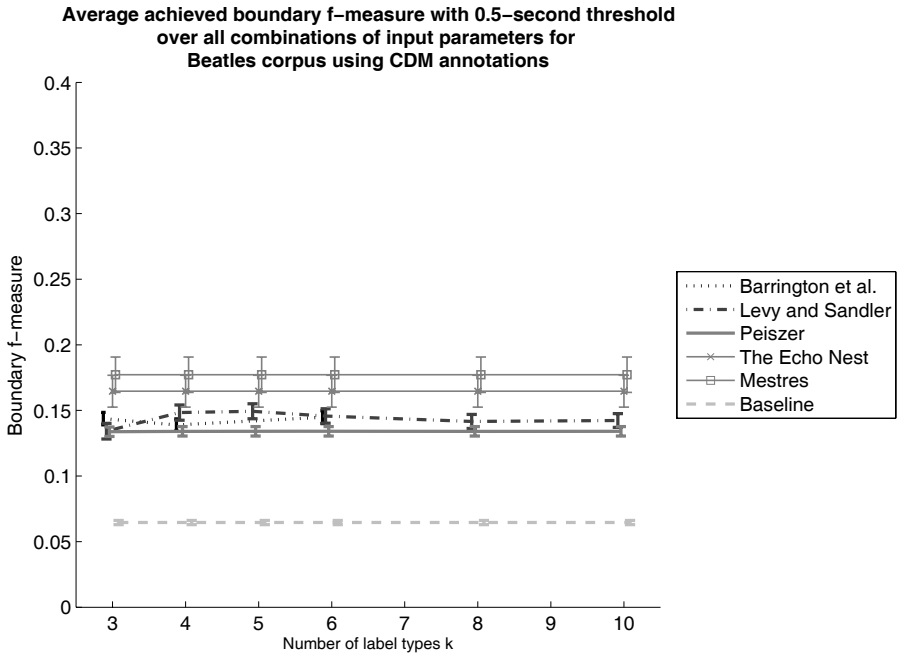
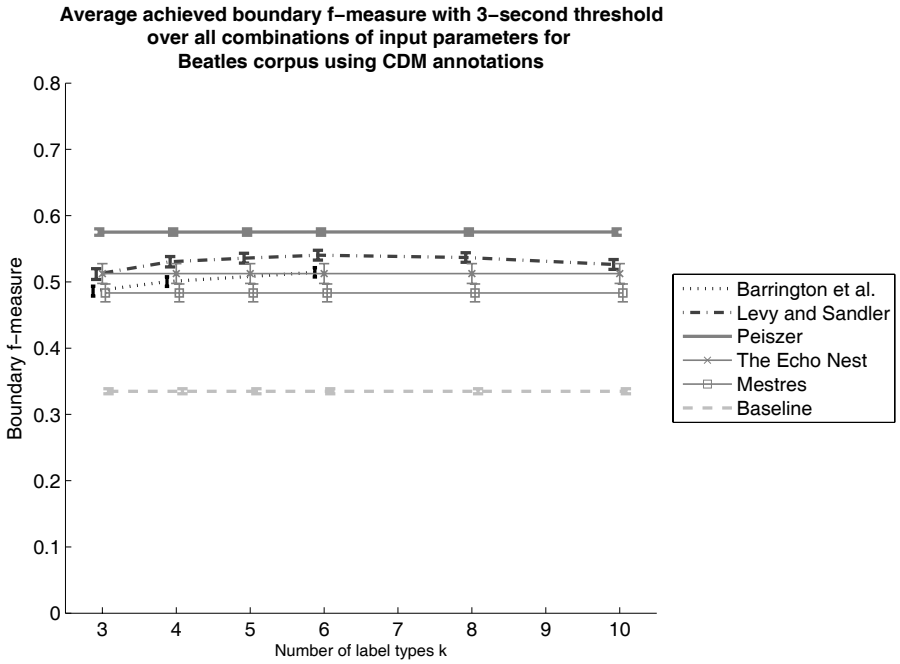

Average achieved Rand index
over all combinations of input parameters
for RWC corpus using RWC annotations
Rand index
Number of label types k
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
Baseline

Figure A.8, a-j. Comparison of the average **boundary *f*-measure** achieved by all input parameter combinations for each algorithm as a function of the specified number of label types *k*. The results are presented for two levels of tolerance, **0.5 and 3 seconds**. Algorithms include Barrington et al., Levy and Sandler, Peiszer, The Echo Nest, Mestres, and the baselines. (Note that The Echo Nest and Mestres' algorithm do not accept an input parameter *k* and therefore their scores are constant over *k*.) The results are presented for each corpus using the original versions of the annotations. Error bars represent standard error.

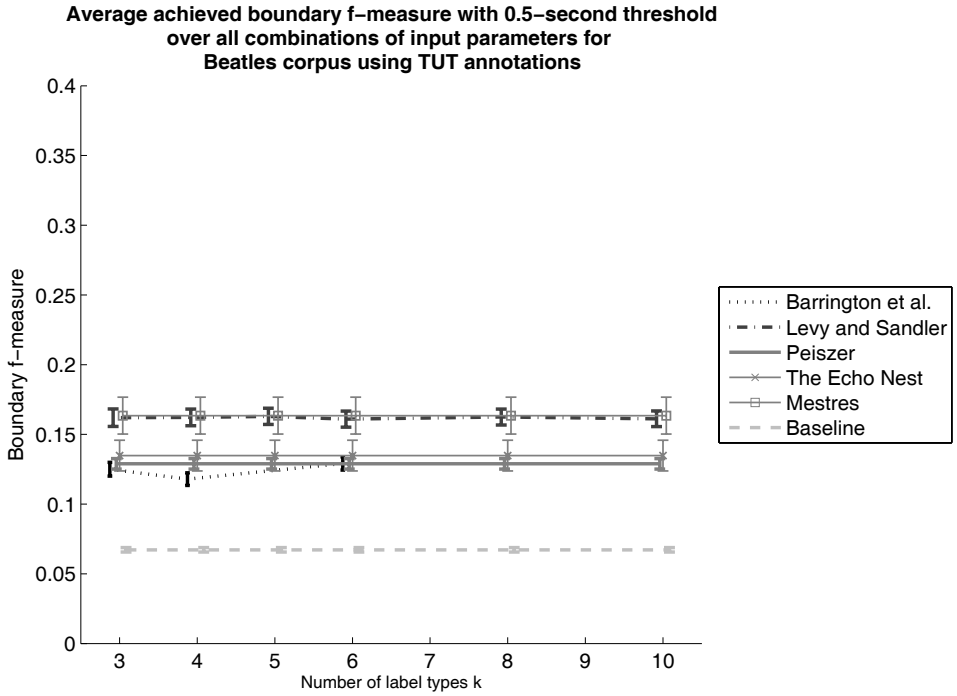
(a)



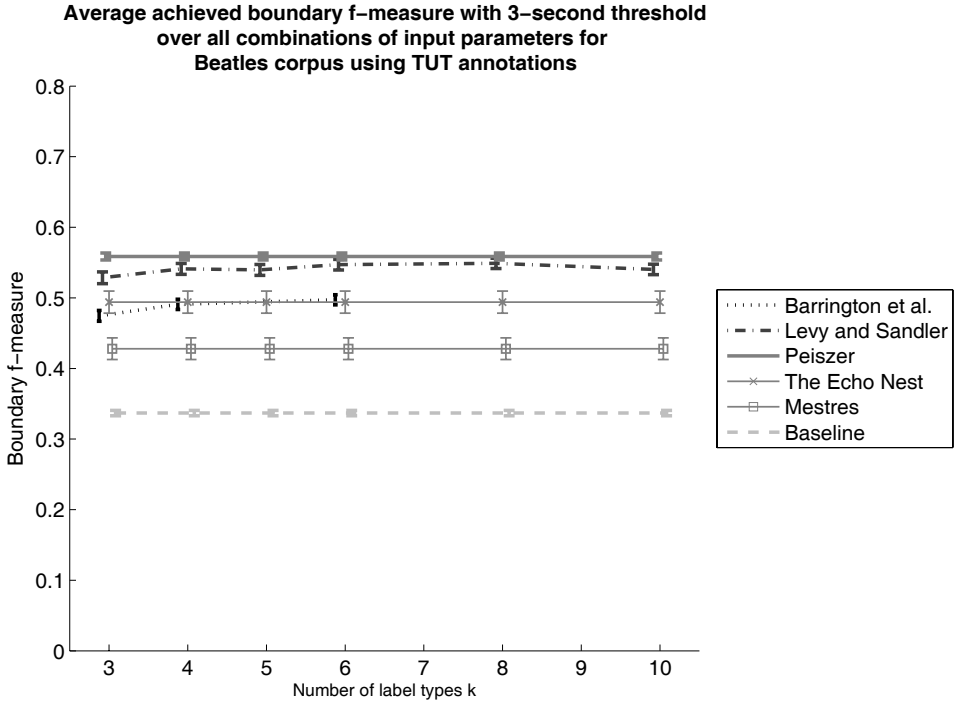

(b)



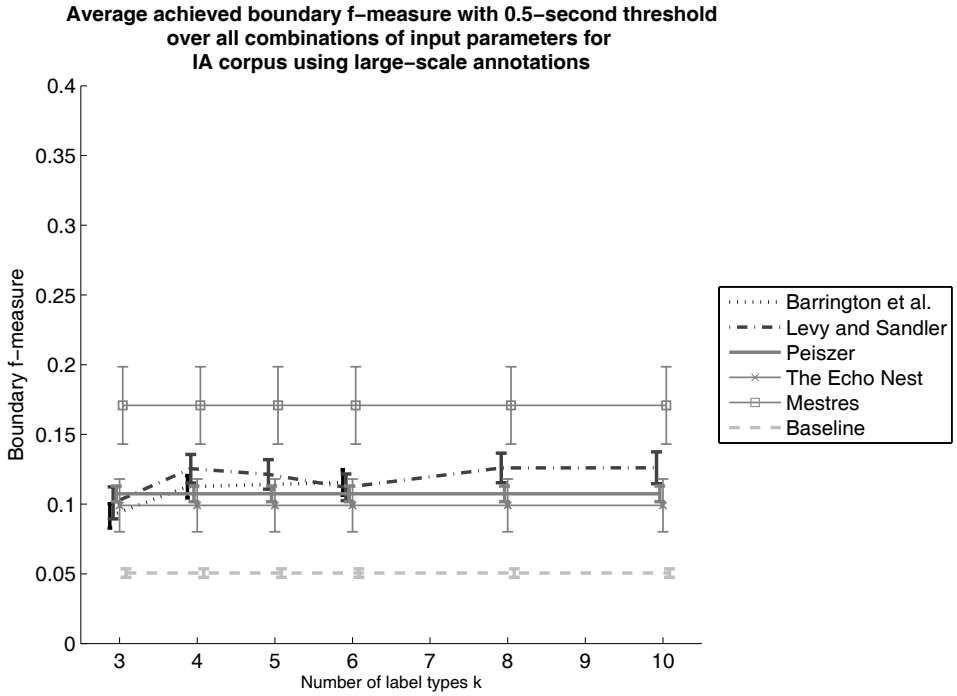
(c)


Average achieved boundary f−measure with 0.5−second threshold
over all combinations of input parameters for
Beatles corpus using TUT annotations
Boundary f−measure
Number of label types k
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

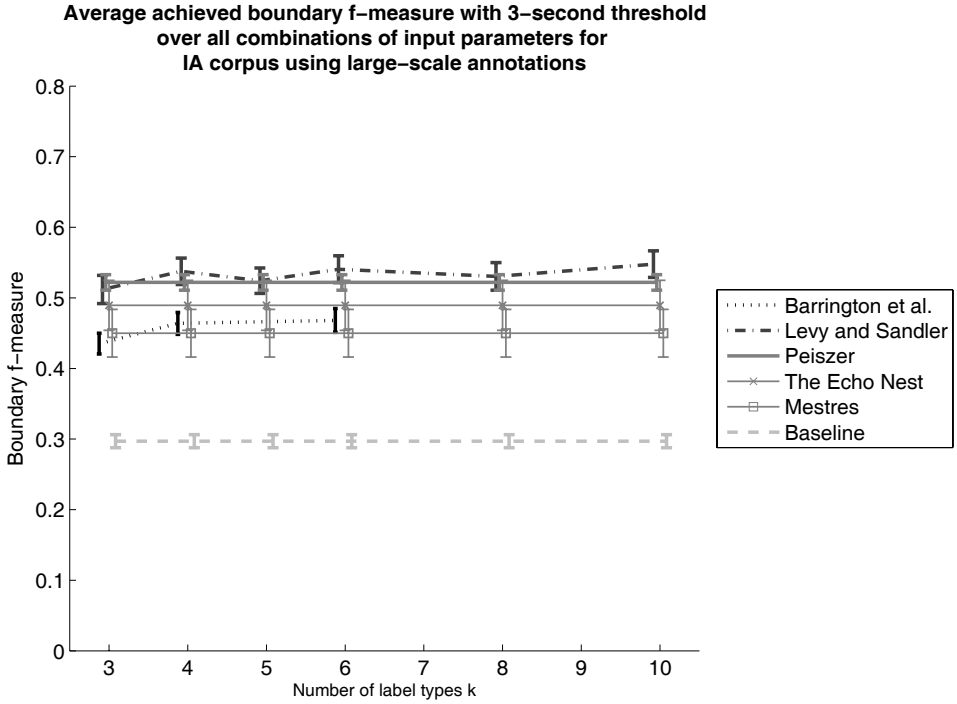
(d)


Average achieved boundary f−measure with 3−second threshold
over all combinations of input parameters for
Beatles corpus using TUT annotations
Boundary f−measure
Number of label types k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

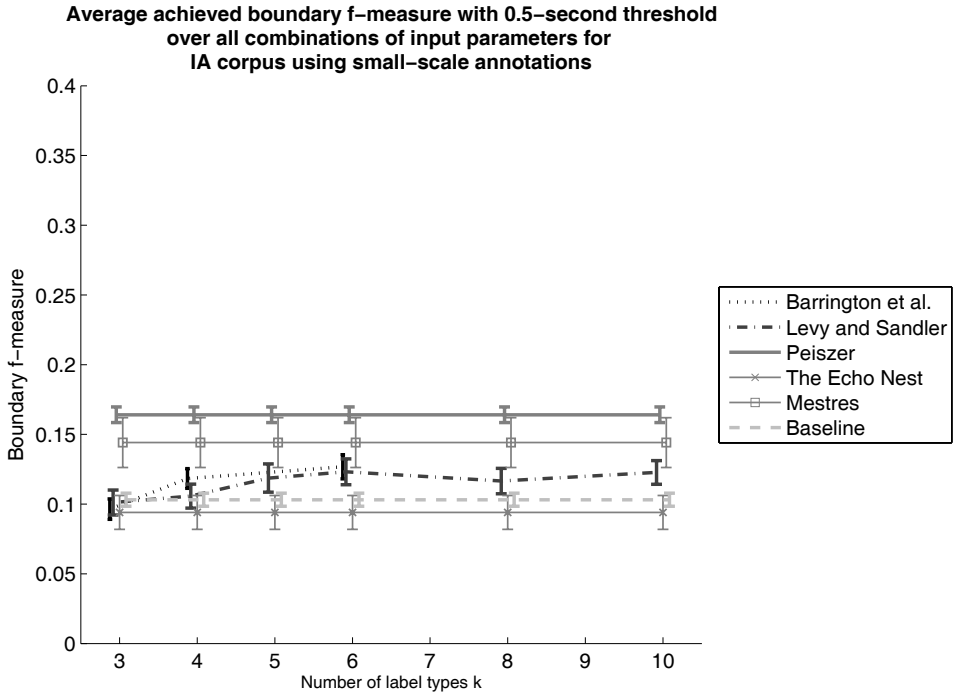
(e)


Average achieved boundary f−measure with 0.5−second threshold
over all combinations of input parameters for
IA corpus using large−scale annotations
Boundary f−measure
Number of label types k
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

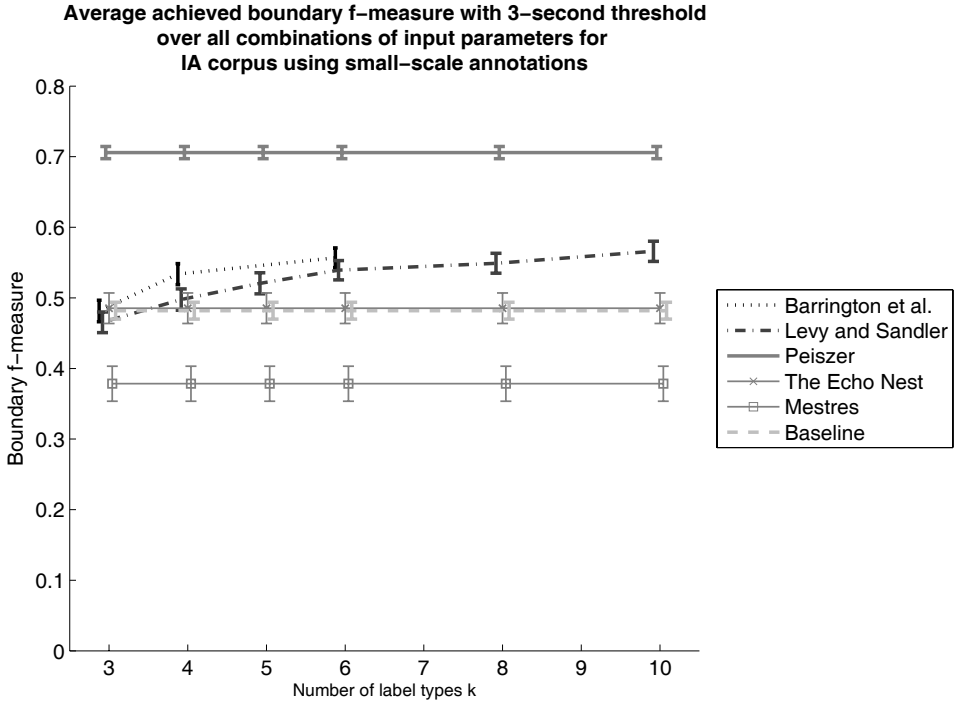
(f)


Average achieved boundary f−measure with 3−second threshold
over all combinations of input parameters for
IA corpus using large−scale annotations
Boundary f−measure
Number of label types k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

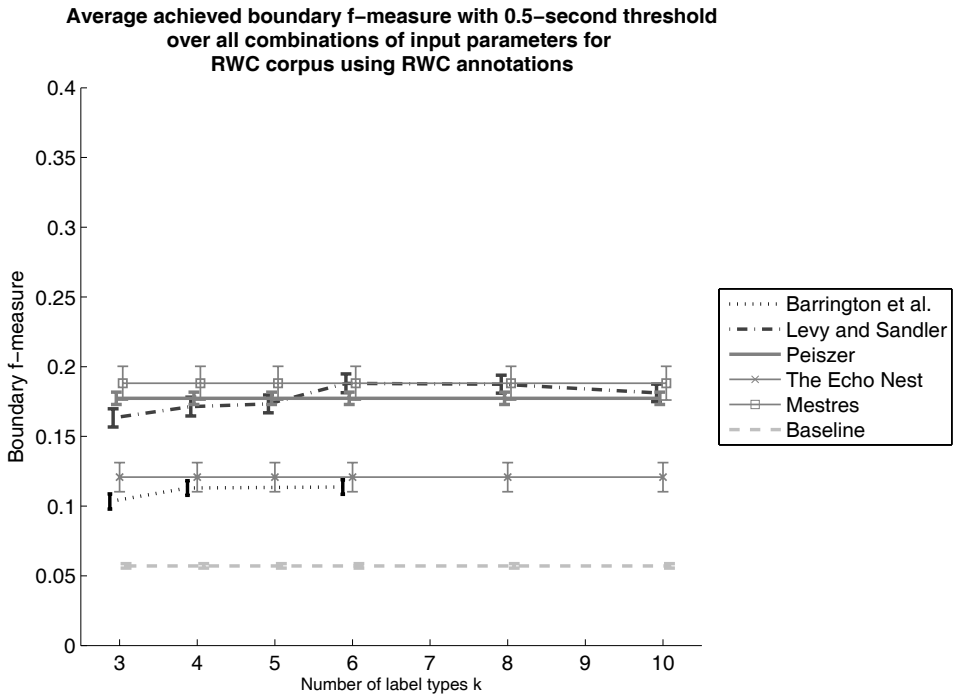
(g)


Average achieved boundary f−measure with 0.5−second threshold
over all combinations of input parameters for
IA corpus using small−scale annotations
Boundary f−measure
Number of label types k
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

(h)


Average achieved boundary f−measure with 3−second threshold
over all combinations of input parameters for
IA corpus using small−scale annotations
Boundary f−measure
Number of label types k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

(i)


Average achieved boundary f−measure with 0.5−second threshold
over all combinations of input parameters for
RWC corpus using RWC annotations
Boundary f−measure
Number of label types k
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

(j)

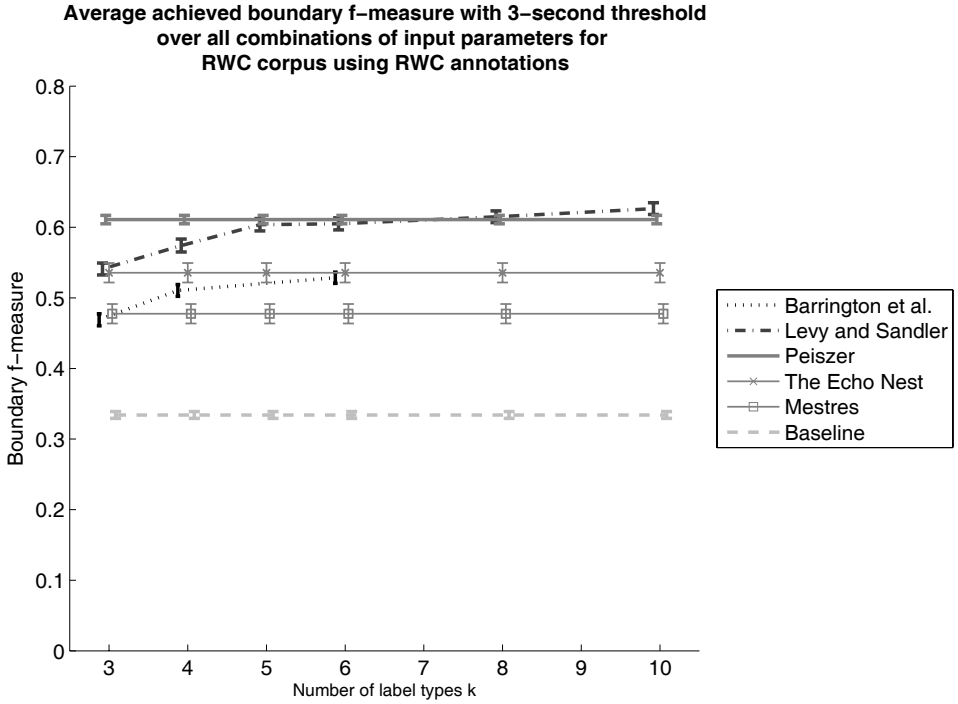
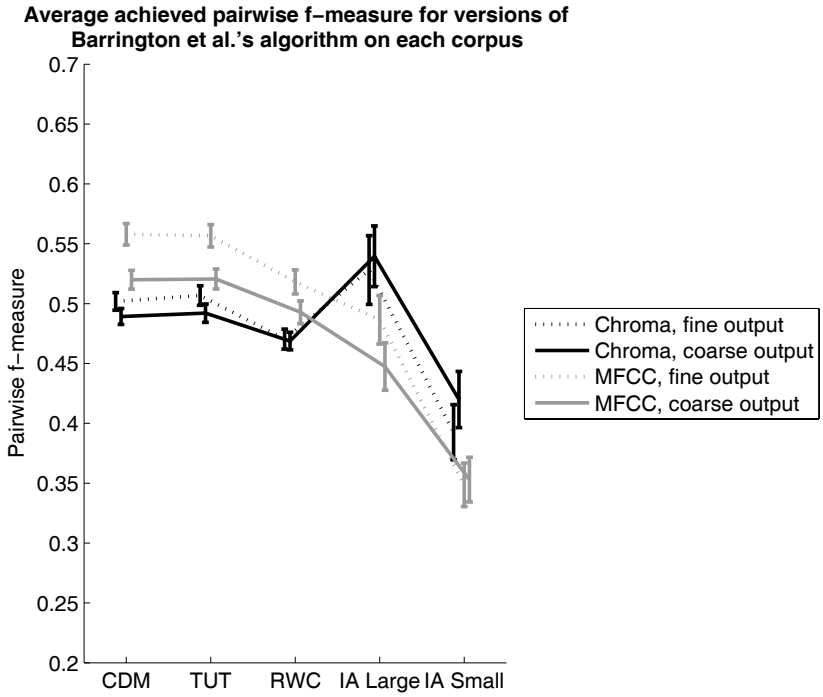
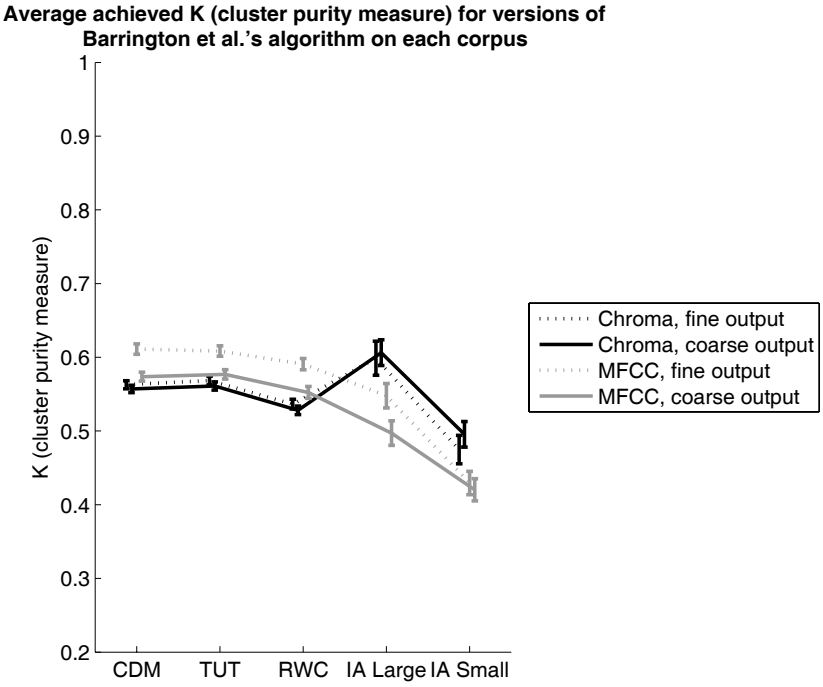

Average achieved boundary f−measure with 3−second threshold
over all combinations of input parameters for
RWC corpus using RWC annotations
Boundary f−measure
Number of label types k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
3
4
5
6
7
8
9
10
Barrington et al.
Levy and Sandler
Peiszer
The Echo Nest
Mestres
Baseline

Figure A.9, a-c. Comparison of the average **pairwise *f*-measure**, **cluster purity measure *K*** and **Rand index**, achieved using each combination of parameters for Barrington et al.'s algorithm, as a function of the corpus of annotations. The original versions of the annotations were used. Error bars represent standard error.

(a)




(b)



(c)

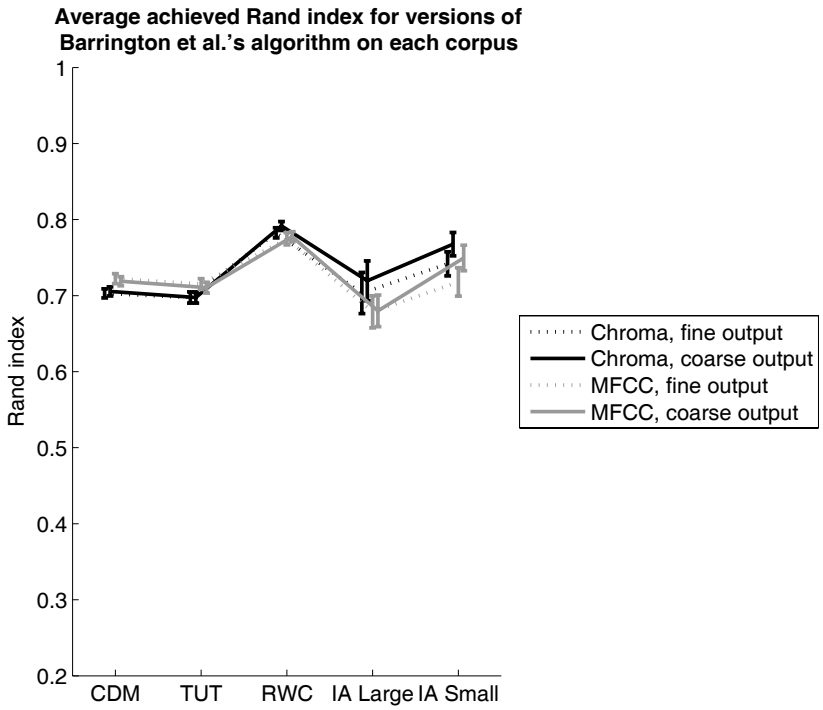
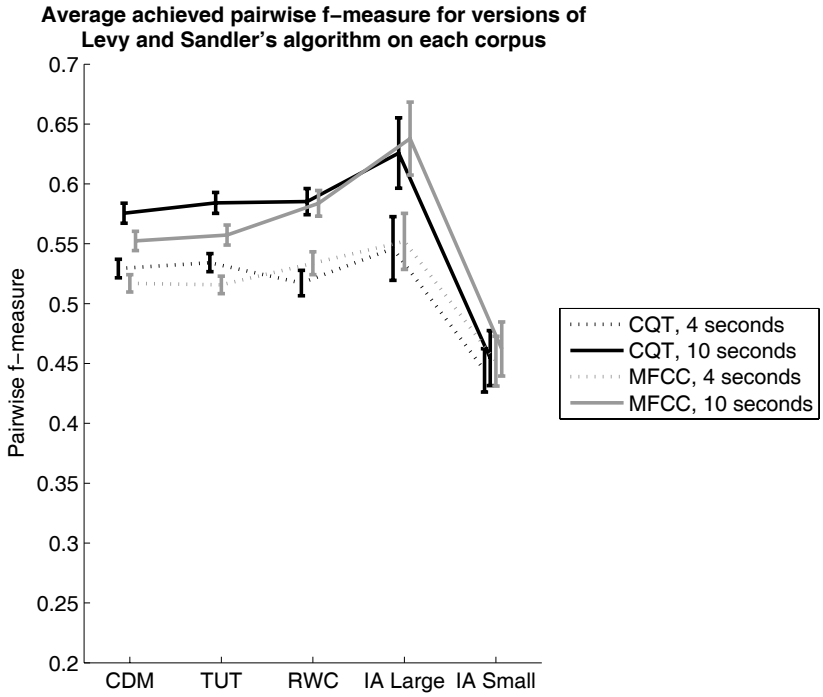
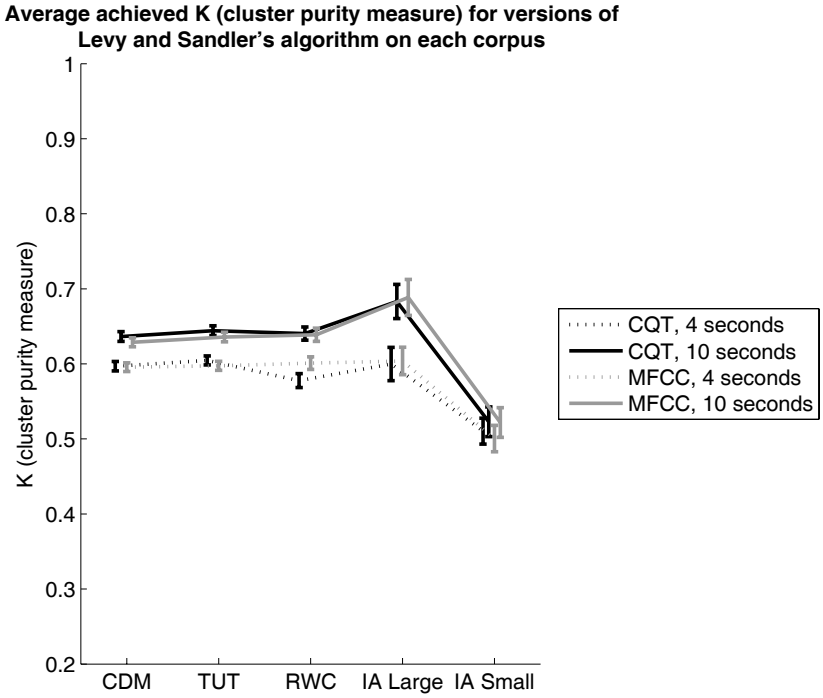

Average achieved Rand index for versions of Barrington et al.'s algorithm on each corpus
Rand index
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
CDM
TUT
RWC
IA Large
IA Small
Chroma, fine output
Chroma, coarse output
MFCC, fine output
MFCC, coarse output

Figure A.10, a-c. Comparison of the average **pairwise *f*-measure**, **cluster purity measure *K*** and **Rand index**, achieved using each combination of parameters for Levy and Sandler's algorithm, as a function of the corpus of annotations. The original versions of the annotations were used. Error bars represent standard error.

(a)




(b)



(c)

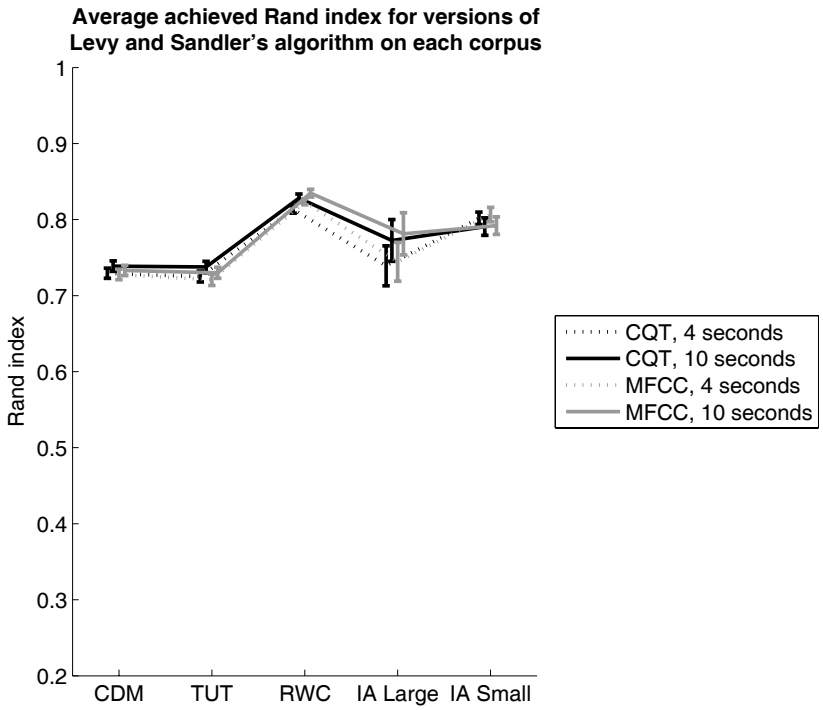
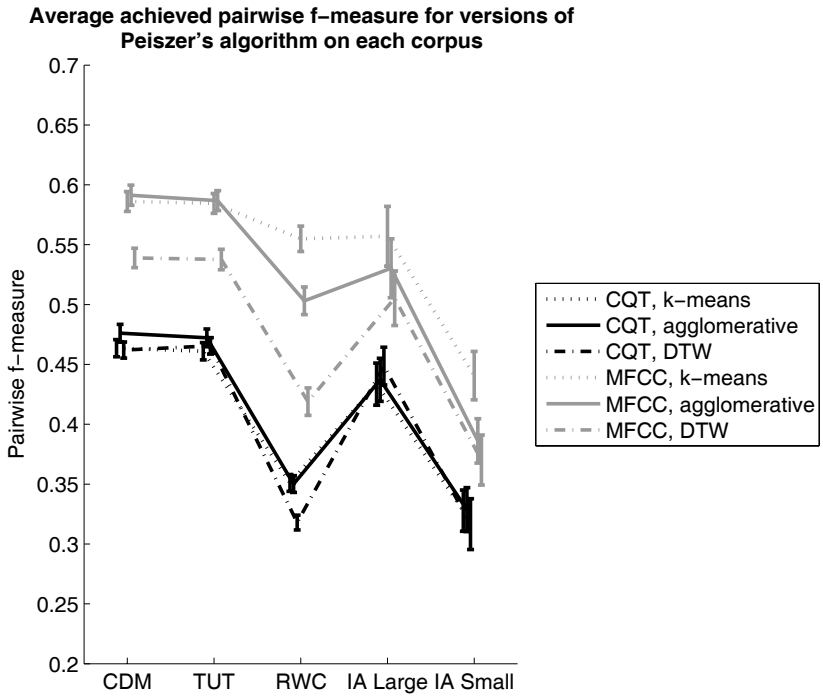
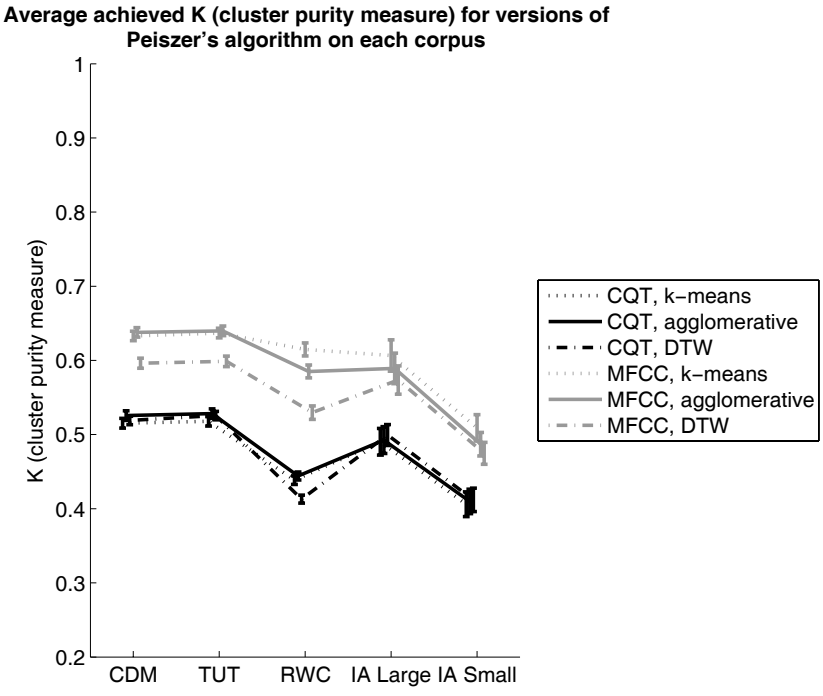

Average achieved Rand index for versions of Levy and Sandler's algorithm on each corpus
Rand index
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
CDM
TUT
RWC
IA Large
IA Small
CQT, 4 seconds
CQT, 10 seconds
MFCC, 4 seconds
MFCC, 10 seconds

Figure A.11, a-c. Comparison of the average **pairwise *f*-measure**, **cluster purity measure *K*** and **Rand index**, achieved using each combination of parameters for Peiszer's algorithm, as a function of the corpus of annotations. The original versions of the annotations were used. Error bars represent standard error.

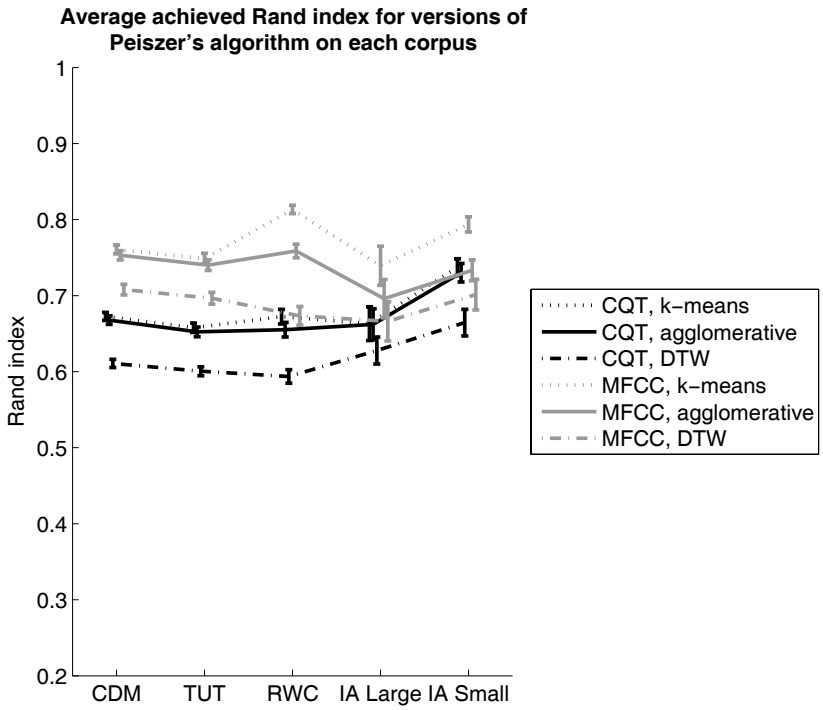
(a)




(b)



(c)


Average achieved Rand index for versions of Peiszer's algorithm on each corpus
Rand index
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
CDM
TUT
RWC
IA Large
IA Small
CQT, k−means
CQT, agglomerative
CQT, DTW
MFCC, k−means
MFCC, agglomerative
MFCC, DTW

# Bibliography


Abdallah, S., K. Noland, M. Sandler, M. Casey, and C. Rhodes. 2005. Theory and evaluation of a Bayesian music structure extractor. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR),* London, UK, 420–5.

Abdallah, S., M. Sandler, C. Rhodes, and M. Casey. 2006. Using duration models to reduce fragmentation in audio segmentation. *Machine Learning* 65 (2–3): 485–515.

Arnold, D., A. Latham, and J. Dunsby. "Form." In *The Oxford Companion to Music*, edited by A. Latham. Oxford Music Online, <http://www.oxfordmusiconline.com/subscriber/article/opr/t114/e2624> (accessed 25 May 2010).

Aucouturier, J.-J. 2001. Segmentation of musical signals, and applications to the analysis of musical structure. Master's thesis, Kings College, University of London, London, United Kingdom.

Aucouturier, J.-J., and M. Sandler. 2001. Using long-term structure to retrieve music: Representation and matching. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Bloomington, IN, United States.

Barrington, L., A. Chan, and G. Lanckriet. 2009. Dynamic texture models of music. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Washington, DC, United States, 1589–92. IEEE Computer Society.

Barrington, L., A. Chan, and G. Lanckriet. 2010. Modeling music as a dynamic texture. *IEEE Transactions on Audio, Speech, and Language Processing* 18 (3): 602–12.

Bartsch, M., and G. Wakefield. 2001. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 15–8.

Bello, J. P. 2009. Grouping recorded music by structural similarity. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR),* Philadelphia, PA, United States, 531–6.

Bimbot, F., O. Le Blouch, G. Sargent, and E. Vincent. 2010. Decomposition into autonomous and comparable blocks: A structural description of music pieces. Technical report, Institut national de recherche en informatique et en automatique (INRIA).

Bruderer, M., M. McKinney, and A. Kohlrausch. 2006. Structural boundary perception in popular music. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 198–201.

Cambouropoulos, E. 1998. Towards a general computational theory of musical structure. Ph. D. thesis, University of Edinburgh, Edinburgh, United Kingdom.

Chai, W. 2003. Structural analysis of musical signals via pattern matching. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Volume 5, 437–40.

Chai, W. 2005. Automated analysis of musical structure. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, United States.

Chai, W., and B. Vercoe. 2003. Structural analysis of musical signals for indexing and thumbnailing. In *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, Washington, DC, United States, 27–34. IEEE Computer Society.

Chan, A., and N. Vasconcelos. 2008. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 30 (5): 909–26.

Cheng, H.-T., Y.-H. Yang, Y.-C. Lin, and H. Chen. 2009. Multimodal structure segmentation and analysis of music using audio and textual information. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Taipei, Taiwan, 1677–80.

de Cheveigné, A. 2006. Multiple $F_0$ estimation. In D. Wang and G. Brown (Eds.), *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*, 45–80. IEEE Press.

Cooper, M., and J. Foote. 2001. Scene boundary detection via video self-similarity analysis. In *Proceedings of the International Conference on Image Processing (ICIP)*, 3: 378–81.

Cooper, M., and J. Foote. 2002. Automatic music summarization via similarity analysis. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 81–5.

Cooper, M., and J. Foote. 2003. Summarizing popular music via structural similarity analysis. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, United States, 127–30.

Cunningham, S., and V. Grout. 2009. Audio compression exploiting repetition (ACER): Challenges and solutions. In *Proceedings of the International Conference on Internet Technologies and Applications (ITA),* Wrexham, North Wales, UK.

Dannenberg, R. 2002. Listening to "Naima": An automated structural analysis of music from recorded audio. In *Proceedings of the International Computer Music Conference (ICMC)*, San Francisco, CA, United States, 28–34.

Dannenberg, R. 2005. Toward automated holistic beat tracking, music analysis, and understanding. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, London, UK, 366–73.

Dannenberg, R., and N. Hu. 2002. Discovering music structure in audio recordings. In C. Anagnostopoulou, M. Ferrand, and A. Smaill (Eds.), *Music and Artificial Intelligence* 2445: 43–57. Springer Berlin / Heidelberg.

Dixon, S. 2001. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research* 30 (1): 39–58.

Dixon, S. 2007. Evaluation of the audio beat tracking system BeatRoot. *Journal of New Music Research* 36 (1): 39–50.

Downie, J. 2008. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology* 29 (4): 247–55.

Eck, D., and J. Schmidhuber. 2002. Learning the long-term structure of the blues. *Lecture Notes in Computer Science 2415*: 796–801.

Eckmann, J.-P., S. Kamphorst, and D. Ruelle. 1987, November. Recurrence plots of dynamical systems. *Europhysics Letters* 4 (9): 973–7.

Eronen, A. 2007. Chorus detection with combined use of MFCC and chroma features and image processing filters. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Bordeaux, France, 229–36.

Eronen, A. 2009. Signal processing methods for audio classification and music content analysis. Ph. D. thesis, Tampere University of Technology, Tampere, Finland.

Fazekas, G., and M. Sandler. 2007. Intelligent editing of studio recordings with the help of automatic music structure extraction. In *Proceedings of the Audio Engineering Society Convention (AES)*, Vienna, Austria.

Foote, J. 1999. Visualizing music and audio using self-similarity. In *Proceedings of the ACM International Conference on Multimedia*, New York, NY, United States, 77–80.

Foote, J. 2000a. ARTHUR: Retrieving orchestral music by long-term structure. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR),* Plymouth, MA, United States.

Foote, J. 2000b. Automatic audio segmentation using a measure of audio novelty. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, 452–5.

Foote, J., and M. Cooper. 2003. Media segmentation using self-similarity decomposition. In M. Yeung, R. Lienhart, and C.-S. Li (Eds.), *Proceedings of the SPIE: Storage and Retrieval for Media Databases*, Volume 5021, Santa Clara, CA, United States, 167–75.

Foote, J., M. Cooper, and A. Girgensohn. 2002. Creating music videos using automatic media analysis. In *Proceedings of the ACM International Conference on Multimedia*, New York, NY, United States, 553–60.

Foote, J., and S. Uchihashi. 2001. The beat spectrum: A new approach to rhythm analysis. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, Los Alamitos, CA, United States, 224–7.

Fremerey, C. 2006. SyncPlayer: a framework for content-based music navigation. Master's thesis, University of Bonn, Bonn, Germany.

Gómez, E. 2006. Tonal description of music audio signals. Ph. D. thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Gómez, E., B. Ong, and P. Herrera. 2006. Automatic tonal analysis from music summaries for version identification. In *Proceedings of the Audio Engineering Society Convention (AES)*, San Francisco, CA, United States.

Goodwin, M., and J. Laroche. 2004. A dynamic programming approach to audio segmentation and speech/music discrimination. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Montreal, QC, Canada, 309–12.

Goto, M., H. Hashiguchi, T. Nishimura, and R. Oka. 2002. RWC Music Database: Popular, classical, and jazz music databases. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 287–8.

Goto, M. 2003a. A chorus-section detecting method for musical audio signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5: 437–40.

Goto, M. 2003b. SmartMusicKIOSK: Music listening station with chorus-search function. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 31–40.

Goto, M. 2006a. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech & Language Processing* 14 (5): 1783–94.

Goto, M. 2006b. AIST annotation for the RWC Music Database. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, 359–60.

Hainsworth, S. 2006. Beat tracking and musical metre analysis. In A. Klapuri and M. Davy (Eds.), *Signal Processing Methods for Music Transcription*, 101–29. New York, NY: Springer.

Hirata, K., S. Tojo, and M. Hamanaka. 2007. Techniques for implementing the Generative Theory of Tonal Music (tutorial session). In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria.

Izumitani, T., and K. Kashino. 2008. A robust musical audio search method based on diagonal dynamic programming matching of self-similarity matrices. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, PA, United States, 609–13.

Jehan, T. 2004. Perceptual segment clustering for music description and time-axis redundancy cancellation. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 124–7.

Jehan, T. 2005a. Creating music by listening. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, United States.

Jehan, T. 2005b. Hierarchical multi-class self similarities. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, United States, 311–4.

Jensen, K. 2005. A causal rhythm grouping. In U. Wiil (Ed.), *Computer Music Modeling and Retrieval*, Volume 3310 of *Lecture Notes in Computer Science*, 83–95. Springer Berlin / Heidelberg.

Jensen, K., J. Xu, and M. Zachariasen. 2005. Rhythm-based segmentation of popular Chinese music. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, London, UK, 374–80.

Kim, S., S. Kim, S. Bong Kwon, and H. Kim. 2006. A music summarization scheme using tempo tracking and two stage clustering. Poster presented at the *IEEE International Workshop on Multimedia Signal Processing (MMSP)*.

Kirlin, P. 2009. Using harmonic and melodic analyses to automate the initial stages of Schenkerian analysis. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 423–8.

Lee, K., and M. Cremer. 2008. Segmentation-based lyrics-audio alignment using dynamic programming. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, PA, United States, 395–400.

Lerdahl, F., and R. Jackendoff. 1983. *A generative theory of tonal music*. Cambridge, MA: MIT Press.

Levy, M., K. Noland, and M. Sandler. 2007. A comparison of timbral and harmonic music segmentation algorithms. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, HI, United States.

Levy, M., and M. Sandler. 2006. New methods in structural segmentation of musical audio. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Florence, Italy.

Levy, M., and M. Sandler. 2008. Structural segmentation of musical audio by constrained clustering. *IEEE Transactions on Audio, Speech, and Language Processing* 16 (2): 318–26.

Levy, M., M. Sandler, and M. Casey. 2006. Extraction of high-level musical structure from audio data and its application to thumbnail generation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Volume 5.

Logan, B., and S. Chu. 2000. Music summarization using key phrases. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Washington DC, United States, 2: 749–52.

Lu, L., M. Wang, and H.-J. Zhang. 2004. Repeating pattern discovery and structure analysis from acoustic music data. In *Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*, New York, NY, United States, 275–82.

Lu, L., and H.-J. Zhang. 2003. Automated extraction of music snippets. In *Proceedings of the ACM International Conference on Multimedia*, New York, NY, United States, 140–7.

Lukashevich, H. 2008. Towards quantitative measures of evaluating song segmentation. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 375–80.

Maddage, N., C. Xu, M. Kankanhalli, and X. Shao. 2004. Content-based music structure analysis with applications to music semantics understanding. In *Proceedings of the ACM International Conference on Multimedia*, New York, NY, United States, 112–9.

Marolt, M. 2006. A mid-level melody-based representation for calculating audio similarity. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 280–5.

Marolt, M. 2009. Probabilistic segmentation and labeling of ethnomusicological field recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 75–80.

Marsden, A. 2007. Automatic derivation of musical structure: A tool for research on Schenkerian analysis. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 55–8.

Martin, B., M. Robine, and P. Hanna. 2009. Musical structure retrieval by aligning self-similarity matrices. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 483–8.

Mauch, M., K. Noland, and S. Dixon. 2009. Using musical structure to enhance automatic chord transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 231–6.

Mermelstein, P. 1976. Distance measures for speech recognition: Psychological and instrumental. In *Proceedings of the Joint Workshop on Pattern Recognition and Artificial Intelligence,* Hyannis, MA, United States, 91–103.

Mestres, X. 2007. A BIC-based approach to singer identification. Master's thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Müller, M., and D. Appelt. 2008. Path-constrained partial music synchronization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, NV, United States, 65–8.

Müller, M., and M. Clausen. 2007. Transposition-invariant self-similarity matrices. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 47–50.

Müller, M., and S. Ewert. 2008. Joint structure analysis with applications to music annotation and synchronization. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, PA, United States, 389–94.

Müller, M., and F. Kurth. 2006. Enhancing similarity matrices for music audio analysis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France, 9–12.

Müller, M., and F. Kurth. 2007. Towards structural analysis of audio recordings in the presence of musical variations. *EURASIP Journal of Applied Signal Processing* (1): 163.

Ong, B. 2005. Towards automatic music structural analysis identifying characteristic within-song excerpts in popular music. Master's thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Ong, B. 2007. Structural analysis and segmentation of music signals. Ph. D. thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Ong, B., E. Gómez, and S. Streich. 2006. Automatic extraction of musical structure using pitch class distribution features. In *Proceedings of the International Workshop on Learning the Semantics of Audio Signals (LSAS)*, Thessaloniki, Greece, 53–65.

Ong, B., and S. Streich. 2008. Music loop extraction from digital audio signals. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, Hannover, Germany, 681–4.

Otsu, N. 1979. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics* 9 (1): 62–6.

Paulus, J. 2009. Signal processing methods for drum transcription and music structure analysis. Ph. D. thesis, Tampere University of Technology, Tampere, Finland.

Paulus, J., and A. Klapuri. 2006. Music structure analysis by finding repeated parts. In *Proceedings of the ACM Workshop on Audio and Music Computing Multimedia (AMCMM)*, New York, NY, United States, 59–68.

Paulus, J., and A. Klapuri. 2008a. Acoustic features for music piece structure analysis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Espoo, Finland, 309–12.

Paulus, J., and A. Klapuri. 2008b. Music structure analysis using a probabilistic fitness measure and an integrated musicological model. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, PA, United States, 369–74.

Paulus, J., and A. Klapuri. 2010. Labelling the structural parts of a music piece with Markov models. In S. Ystad, R. Kronland-Martinet, and K. Jensen (Eds.), *Computer Music Modeling and Retrieval: Genesis of Meaning in Sound and Music*, 5493: 166–76. Berlin, Heidelberg: Springer-Verlag.

Peeters, G. 2004. Deriving musical structures from signal analysis for music audio summary generation: "sequence" and "state" approach. In G. Goos, J. Hartmanis, and J. van Leeuwen (Eds.), *Computer Music Modeling and Retrieval*, 2771: 169–85. Springer Berlin / Heidelberg.

Peeters, G. 2007. Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 35–40.

Peeters, G., and E. Deruty. 2009. Is music structure annotation multi-dimensional? A proposal for robust local music annotation. In *Proceedings of the International Workshop on Learning the Semantics of Audio Signals (LSAS)*, Graz, Austria, 75–90.

Peeters, G., A. La Burthe, and X. Rodet. 2002. Toward automatic music audio summary generation from signal analysis. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 94–100.

Peiszer, E. 2007. Automatic audio segmentation: Segment boundary and structure detection in popular music. Master's thesis, Technische Universität Wien, Vienna, Austria.

Peiszer, E., T. Lidy, and A. Rauber. 2008. Automatic audio segmentation: Segment boundary and structure detection in popular music. In Proceedings of the International Workshop on Learning the Semantics of Audio Signals (LSAS), Paris, France, 45–59.

Pollack, A. 2000. Allan W. Pollack's "Notes ... On" series. The "official" rec.music.beatles homepage. <http://www.recmusicbeatles.com/public/files/awp/awp.html> (accessed 10 August 2010).

Rauber, A., E. Pampalk, and D. Merkl. 2002. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 71–80.

Rhodes, C., and M. Casey. 2007. Algorithms for determining and labelling approximate hierarchical self-similarity. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 41–6.

Rhodes, C., M. Casey, S. Abdallah, and M. Sandler. 2006. A Markov-chain Monte-Carlo approach to musical audio segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France, 797–800.

Sandler, M., and M. Levy. 2007. Signal-based music searching and browsing. In *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, United States, 1–2.

Shiu, Y. 2007. Digital signal processing techniques for music structure analysis. Ph. D. thesis, University of Southern California, Los Angeles, CA, United States.

Shiu, Y., H. Jeong, and C.-C. J. Kuo. 2006. Similarity matrix processing for music structure analysis. In *Proceedings of the ACM Workshop on Audio and Music Computing Multimedia (AMCMM)*, New York, NY, United States, 69–76.

Smoliar, S. 1980. A computer aid for Schenkerian analysis. *Computer Music Journal* 4 (2): 41–59.

Stammen, D., and B. Pennycook. 1994. Real-time segmentation of music using an adaptation of Lerdahl and Jackendoff's grouping principles. In *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*, Liege, Belgium, 269–70.

Su, M.-Y., Y.-H. Yang, Y.-C. Lin, and H. Chen. 2009. An integrated approach to music boundary detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 705–10.

Turnbull, D., G. Lanckriet, E. Pampalk, and M. Goto. 2007. A supervised approach for detecting boundaries in music using difference features and boosting. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 51–4.

Tzanetakis, G., and P. Cook. 1999. Multifeature audio segmentation for browsing and annotation. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA),* New Paltz, NY, United States, 103–6.

Van Steelant, D., B. De Baets, H. De Meyer, M. Leman, J. Martens, L. Clarisse, and M. Lesaffre. 2002. Discovering structure and repetition in musical audio. In *Proceedings of the EUROFUSE Workshop on Information Systems*, Verona, Italy, 43–8.

Wellhausen, J., and M. Höynck. 2003. Audio thumbnailing using MPEG-7 low level audio. In J. Smith, S. Panchanathan, and T. Zhang (Eds.), *Proceedings of the SPIE: Internet Multimedia Management Systems*, 5242: 65–73.

Wolkowicz, J., S. Brooks, and V. Keselj. 2009. Midivis: Visualizing music structure via similarity matrices. In *Proceedings of the International Computer Music Conference (ICMC)*, Montreal, QC, Canada, 53–6.

Xu, C., N. Maddage, and X. Shao. 2005. Automatic music classification and summarization. *IEEE Transactions on Speech and Audio Processing* 13 (3): 441–50.

Xu, C., N. Maddage, X. Shao, and Q. Tian. 2007. Content-adaptive digital music watermarking based on music structure analysis. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 3 (1).

Xu, C., X. Shao, N. Maddage, M. Kankanhalli, and Q. Tian. 2004. Automatically summarize musical audio using adaptive clustering. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, Taipei, Taiwan, 2063–6.

Xu, C., Y. Zhu, and Q. Tian. 2002. Automatic music summarization based on temporal, spectral and cepstral features. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, 117–20.

Zhang, T., and R. Samadani. 2007. Automatic generation of music thumbnails. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, Beijing, China, 228–31.

Zhu, Y., K. Chen, and Q. Sun. 2005. Multimodal content-based structure analysis of karaoke music. In *Proceedings of the ACM International Conference on Multimedia*, Singapore, 638–47.