Keywords: audio content analysis, audio semantic analysis, acoustic unit descriptors, structured models for audio, multi-instance learning for audio
Abstract

The ability to automatically analyze audio content is a key aspect of information retrieval systems that deal with multimodal files. The unprecedented growth of web-based user generated content-sharing platforms and their popularity has led to research efforts attempting to understand the content of such files. Typically, audio analysis research has focused on some specific tasks – detection of specific types of sounds, classification of the content into categories, and summarizing the content of an audio file. These approaches involved working individually on small segments of audio using supervised methods to detect patterns of interest.

The main hypothesis that drives this dissertation is that sound has its own language and structure and can be modeled using sequences of lower level units (which we refer to as acoustic unit descriptors). The lower level units may not carry semantic information individually, but the sequences or distribution of these units should capture semantic information. In this language for sounds, the lower level units alluded to would be analogous to the alphabet.

Such a representation of sound using a discrete sequence lends itself naturally to a hierarchical structure, where sequences of these lower level units can be mapped to real events, that have clear semantic interpretations. Further, these event sequences themselves should carry information about the overall semantic content or class of the audio. Depending on the restrictions we enforce at various levels of this structure, we can use such structured models to classify or detect sound types, segment files as a sequence of semantically meaningful sound types, or predict associated sound classes.

In this proposal, we first summarize our prior work that describes the process of learning of the lower level acoustic unit descriptors in an unsupervised manner from audio data. We demonstrate empirically that the learnt acoustic unit descriptors appear to capture semantic information, and that they can outperform other plausible semantically motivated schemes.

We then discuss the proposed directions of research in this dissertation, including techniques that attempt to discover further structural relationships between sequences of these acoustic unit descriptors, or the event layer that lies above them. Our approach to discovering the hidden structure proposes to leverage the large amount of unlabeled and coarsely labeled data, using techniques inspired by semi-supervised and multi-instance learning approaches.

The research pursued in this dissertation will demonstrate that hidden semantic structure can be automatically discovered from weakly-labeled audio data. The use of such semantically informed features would enable audio analysis to improve significantly over the state-of-the-art.
## List of Figures

1.1 An example of the levels of analysis involved. The color *blue* is used for data or human annotations, and *black* for the labels that should be generated by our system. 2  
2.1 Example of a syntactic parse of a sentence (from Charniak [1997]) .......................... 7  
2.2 Example of a dependency parse of a sentence (from McDonald and Pereira [2006]) 7  
3.1 Example of a sequence from a baseball video. .......................................................... 12  
3.2 Graphical model for each data point D. Circles represent random variables and rectangles represent parameters. ................................................................. 14  
3.3 AUC for the various feature sets (lower is better) ..................................................... 22  
3.4 ROC with AUDs$_{\text{binary}}$ .......................................................... 22  
3.5 ROC with AUDs$_{\text{framecount}}$ .......................................................... 22  
3.6 ROC with AUDs$_{\text{count}}$ .......................................................... 22  
3.7 A plot of frequency sorted AUDs from our initialization, which appears to follow a power law. ................................................................. 28  
3.8 A plot of the frequency sorted AUDs at different points in the learning process for AUDs. ................................................................. 28  
3.9 Changes in the unigram distribution of AUDs over time ............................................. 29  
3.10 Changes in the bigram distribution of AUDs .............................................................. 29  
4.1 An example of parsing of text data. The color *black* represents the observed data and *green* represents inferred structure (Source: Wikipedia) .......................... 31  
4.2 An example of a simple parse tree over AUDs. .......................................................... 32  
4.3 An example parse tree for baseball. We assume that the complex events ($e_i$) do not contain dependencies ................................................................. 32  
4.4 Dependency tree for a sequence in a movie. The orange edge represents an alternative tree where the event $v_5$ is dependent on $v_1$ instead of $v_4$, resulting a non-projective tree. ................................................................. 33  
4.5 The joint model over the grammar, AUDs, and observed audio. .................................. 36  
4.6 A decoupled model with the process of generation of AUDs with the grammar separated from the generation of observed audio with AUDs. .................................. 36  
4.7 An example of a dataset for multi-instance learning .................................................. 38
## List of Tables

3.1 Classification based on Viterbi decoding scores .......................... 18
3.2 Classification using MIRA classifier ............................................ 18
3.3 Category specific accuracy for the various classes .......................... 18
3.4 Results on MED10 test data ....................................................... 27

4.1 Estimated timeline for progress toward thesis completion ................... 40
Chapter 1

Introduction

Sound has a profound effect on our perception of the world around us. It affects our sense of
beauty, level of alertness, ability to focus, and often mental state. While various acoustic phe-
nomena, such as speech recognition, language identification, music classification, have been well
studied, a significant portion of the knowledge that would help advance the area of automatic
understanding still elude us. Even though some of these aspects such as computer aided mu-
sic, content-based retrieval of audio are the subject of significant research efforts, approaches
toward developing a more general understanding of sound—relationships between various sound
types, semantic links between sound events, discovery of sub-event structures—remain largely
unexplored.

The world around us is, however, structured in space and time, and the evolution of naturally
occurring phenomena with time is related to the previous states. The changes that occur in any
given scene in the real world are sequential by nature and the human brain can perceive and
understand the sequential nature of these changes, as well as the semantic relationships between
the various events. e.g. the movement of traffic and people at an intersection are governed by
the traffic laws and changes in the traffic signals; a driver will sound his horn to warn another
driver or a pedestrian. While the semantic information linking event sequences exists in both the
visual and audio modalities, we believe that the audio alone carries significant information, and
the automatic discovery of semantic structure from audio is the focus of this dissertation.

Automatic analysis of audio content is a key aspect of information retrieval systems Foote
[1997], Wold et al. [1996] that deal with multimodal files. Specifically, for the purposes of in-
dexing and classification, audio analysis research has focussed on specific tasks – detection of
specific sounds, classification of the audio into categories, retrieval of documents in response to
queries. These approaches primarily involved working individually on small segments of audio
using supervised methods to detect patterns of interest.

The main hypothesis that drives this dissertation is that sound has its own language and
structure, especially with respect to its semantic content. A sound file typically consist sof a
sequence of semantically meaningful acoustic events that are usually related to each other, and
sound context provides valuable cues that can aid analysis of sound. This leads us to an analysis
framework where we seek to discover event sequences, and the structural relationships between
individual events in such sequences instead of attempting to detect interesting patterns on isolated
segments.

Further, we expect that these acoustic events themselves are composed of lower level units
that carry semantic information, and directly govern the observed acoustics. Thus, sound can be
modeled using sequences of lower level units (which we refer to as acoustic unit descriptors), which may not carry discernible semantic information individually, but the sequences or distribution patterns of these units should capture coherent semantic information (acoustic events. In this language for sounds, the lower level units alluded to would be analogous to the alphabet.

Such a representation of sound lends itself naturally to a hierarchical structure as shown in Fig 1.1. Further, these event sequences themselves should carry information about the overall semantic content or class of the audio. Sequences or patterns over AUDs should represent semantically meaningful events (represented by $E_i$ in the figure). Further, in most natural audio, these events themselves do not occur in isolation. Various events are related to each other in different ways, and event context should provide cues for possible future events (event dependencies are indicated by arrows in the figure).

A hierarchical analysis structure can be exploited for various common tasks in audio analysis, as well as to generate a better understanding of the semantic relationships between audio events. This dissertation aims to develop methods that can model audio using the hierarchical structure described earlier. The primary issue that arises in our setting for semantic analysis of audio is a scarcity of richly annotated data with information at various hierarchical levels that could be used in supervised settings. To address this issue, we propose to use easily available data that only contain weak or no supervision, and perform learning in unsupervised or weakly supervised settings. The dissertation will demonstrate that sufficient information can be extracted using such methods to allow semantic analysis of audio data.

While an ideal framework for evaluation would directly measure the accuracies of the generated structure by comparing with ground truth structures (generated by annotators), such data is expensive to obtain. We propose, instead, to use the structured information generated by our
models as features that would provide improved information for various common audio processing applications.

I Audio Classification and Retrieval: Given an audio file, the task of audio classification requires us to assign the entire audio file to one of multiple pre-defined classes. Depending on the set of pre-defined classes, such a system would be useful for tasks such as music genre identification and retrieval of specific classes of documents from a large collection.

II Audio Segmentation: Given an audio file, the segmentation task requires us to find (semantically) coherent segment boundaries automatically. Note that in Figure 1.1, such a segmentation is naturally available from the events layer ($E_i$). The ability to simply detect boundaries without identifying them allows one to obtain smaller (and potentially, simpler) sequences for other tasks such as detection and subsequent recognition of speech, classification of segments into semantic classes. A joint segmentation and identification of segments provides us with information about segment sequences that can be used for tasks like event recounting or as features for audio classification among others.

III Audio Event Detection: Given a set of audio files, an audio event detection task attempts to locate all occurrences of a particular audio event within the audio files. We expect that a hierarchical analysis will enable us to better understand contexts within which different events occur and use this knowledge to do a better job of detecting events.

Beyond their direct applications in tasks like retrieval and recounting, structured, hierarchical relationships between various sound types, audio events would be interesting to analyze and understand. They can improve our understanding of the sequential (or co-occurrence) relationships of various sound types, e.g., music and whether sequences of audio events (notes, note sequences) can help identify composers, genres, etc.

The rest of this document is organized as follows: in Chapter 2, we review work in various areas that are closely related to our problem of interest. In Chapter 3, we discuss our preliminary work towards inferring structure from audio and its applications in addressing various problems. Chapter 4 describes the proposed future work to be completed as part of this dissertation, along with a brief description of the proposed techniques. We conclude with a discussion of our contributions in this thesis and a timeline for this dissertation.
Chapter 2

Related Work

This chapter presents a summary of some research areas that are relevant to this thesis. Automatic content analysis in multimodal data involves two main areas of research emphasis—the analysis of the audio and the video component, which often carry complementary information to each other. Even though the focus of this thesis is on audio analysis, various aspects of research in the video domain bear similarities to our formulation of the problem, and to one of the main problems we encounter—learning from weakly supervised data.

The hierarchical structured framework described earlier bears significant resemblance to the structures that text parsing approaches have sought to infer. While audio differs in quite a few significant ways from text, we can build on certain frameworks that have been shown to be good for parsing natural language text.

Finally, we briefly discuss approaches that have been developed for learning in settings where very little supervised data is available, and would be expensive to obtain. The rich hierarchical structure we propose to infer from audio does not exist in any standard audio datasets. Further, while various audio data sets do contain informative labels, they often apply only at specific (and usually unknown) granularities. The ability to infer labels from unlabeled data using small amounts of labeled data, as well as understanding the granularities at which such labels apply, automatically would make audio analysis systems more powerful. To this end, we propose to leverage methods from machine learning approaches relating to semi-supervised learning.

In this chapter, we review relevant work from these areas, and discuss similarities and differences with our work and approaches. Section 2.1 discusses related work in content-based audio processing, section 2.2 discusses methods used for parsing of natural language text, and Section 2.3 discusses some relevant areas of machine learning research. Finally, in Section 2.4, we draw some parallels between work in the image processing community and our work.

2.1 Content-based Audio Processing

The rapid increase in popularity of web-based systems that allow users to share data on the web has resulted in an unprecedented increase of multimodal content on the internet. Automatic analysis of the content of these files is essential in order to be able to index and retrieve relevant files in response to user queries. The audio in the multimedia files provides significant information regarding the semantic content to human users. An obvious solution to the problem of handling large amounts of audio data is to annotate it with textual information and then use traditional IR techniques for searching. This approach works well and has the advantage of using well-
understood techniques. On the other hand, using current interfaces for human annotation of audio is extremely time-consuming. As a result, there has been significant research effort attempting to automatically utilize the audio content for indexing of multimedia files.

Content based audio classification and retrieval is essentially a 2 step pattern recognition problem– first, the audio is represented using a set of features. These features are then used for classification. The earliest efforts Liu et al. [1997], Wold et al. [1996] sought to match perceptual features of audio files to an audio query.

As speech recognition systems improved, the task of spoken document retrieval using text queries on large speech corpora Garofolo et al. [1997] was well studied. The standard approach used by systems in this setting involve transcribing the audio signal, and using text-based retrieval techniques on the transcribed text. Systems that could perform keyword spotting Szoke et al. [2005] and query by example Velivelli et al. [2004] were also developed for retrieval from these large speech databases.

The task of understanding non-speech sounds, however, is a much harder one. In settings where the domain of sounds is unconstrained (or audio in the wild, as it is often referred to), one can imagine an infinite number of potential sounds. As a result, approaches to audio processing in this area were task-driven– e.g. detecting specific sounds in audio using detectors that sought to classify chunks of audio as containing the target sound or not. The most common approach is to use a vocabulary of sounds, comprising clearly characterizable sounds such as gunshots, laughter, speech, animal sounds, music, crowd sounds etc. Audio is analyzed by detecting the presence of sounds from this vocabulary in it and additional analysis builds on top of such detection. For instance, Chang et al. [2007] identify the presence of sounds from a vocabulary and combine this information with evidence from video. Slaney [2002a] describes a system that could be used to map between regular vocabulary and sounds by association. Friedland et al. [2009] navigates Seinfeld episodes taking advantage of traditional sitcom artifacts, such as music indicating scene changes and laughter following punchlines. Other analyses detect repeated sequences in a television broadcast stream Berrani et al. [2008], with the intent of identifying jingles, advertisements and so forth. Many of these methods work well in restricted domains, and based on these techniques, an unconstrained, completely automatic system for audio understanding can be envisioned.

Semantic analysis of audio has also been explored recently in multimodal settings. Jiang et al. [2009] utilize the concept of Short-Term Audio Visual Atoms (S-AVA) where features are extracted from both the video and the audio and are used together to develop codebooks for various semantic concepts. The codewords in the codebooks are then used as features to train classifiers to detect the concept. Lee and Ellis [2010] used a set of 25 semantic classes (e.g. dancing, singing, birthday) for classifying consumer video clips using only audio information. Rui et al. [2000] attempted to generate highlights for baseball audio using information from the audio track only using energy-based features, as well as phoneme-level features and prosodic features from the announcers’ speech. Divakaran et al. [2003] used audio features in conjunction with video features based on motion activity for news video summarization– they used Hidden Markov Models for modeling various kinds of audio events such as speech, barking, etc. which were then used to segment the sound track, as well as detect speakers and speaker changes.

In the above, the basic mechanism involves spotting a set of known sound types in audio. Higher-level descriptions of audio must be obtained by further inference or by human supervision, after the sounds are detected. Our work takes a different approach. We model all audio as being composed of a sequence of a relatively small set of atomic sound units. The contents of an audio
recording are represented by the specific sequence of units that compose it. All recordings of any category of sound are have similar compositions in terms of these units, and a hierarchical structure including event information can be deduced automatically, as shown in Fig 1.1.

Besides the annotation errors arising out of misclassifications, Tzanetakis and Cook [1999] raise the issue of subjectivity of the listener in assigning annotations to audio, and suggest an interactive framework that combines manual and automatic annotations into a flexible, unified framework. Our framework is equipped to handle such situations, as well. We will describe in Section 3.2.4 a method that can be adapted to take into account individual preferences in generating labels for segments of audio that can easily be personalized for different users.

While techniques for analysis of general audio does not typically pay attention to structure, there has been a great deal of work on using grammars for generating or analyzing music. Early approaches to grammar based generation of music employed simple, deterministic rule-based techniques Steedman [1989]. Researchers have attempted to use various kinds of grammars for different tasks– tree grammars were used to compute melodic similarity computation and melody classification Bernabeu et al. [2011], analysis of musical structure to recover the sectional form of a musical piece using MFCC, chroma and rhythmogram features Paulus and Klapuri [2009], and using regular grammars to model musical style for classification Cruz-Alcazar and Vidal [2008].

### 2.2 Natural Language Parsing

Significant research efforts have been made towards the goal of automatically parsing natural language text to predict the syntactic structure of the text. The process of parsing enables the identification of the structure of meaningful subsequences of any text sentence, including phrase boundaries spanning sets of words and identification of the parts-of-speech for the individual words, producing a syntactic structure as shown in Figure 2.1 Charniak [1997]. It is generally accepted that the parse tree obtained in this manner is useful in understanding the sentence automatically, and features derived from such parse trees are used for a variety of applications, including question-answering, machine translation, textual entailment.

The current state-of-the-art systems for natural language parsing all employ statistical methods to determine the most likely syntactic structure for any given sentence. From the parse tree shown in Figure 2.1, one can see that words are first categorized into parts of speech, and phrase boundaries are obtained over sequences of words (in the Figure, NP refers to a noun phrase, and VP to a verb phrase). Finally, the sentence is a sequence of such phrases– note that a phrase can span sub-phrases as well. Consider a sentence such as ”The dog ate the biscuit”. In the parse for this sentence, ”the biscuit” would have parts of speech ”det noun”, which would be spanned by an NP. Then the VP would span ”verb NP”, thus spanning the words ”ate the biscuit”.

Openly available part-of-speech taggers include the Stanford POS Tagger Toutanova et al. [2003], and the CRF Tagger Phan [2006], and implement various methods ranging from Hidden Markov Model formulations Rabiner and Juang [1986] to Maximum Entropy techniques Toutanova and Manning [2000] and Conditional Random Fields Lafferty et al. [2001].

As a precursor to full parsing, many researchers worked on the task of chunking Abney [1991], Ramshaw and Marcus [1995], which involved dividing sentences into non-overlapping segments based on analysis of the words over windows. However, statistical parsing techniques are currently sufficiently developed to generate full parses for sentences of the kind shown in Figure 2.1. Again, statistical parsers for full parsing of natural language sentences employ a variety of models including probabilistic context free grammars and maximum entropy models.
More recently, dependency-based methods for syntactic parsing have increased in popularity. A comprehensive survey of such methods is available in Nivre [2006] and Kubler et al. [2009]. A dependency parse of a sentence produces a dependency graph such that relationships between words and their arguments are represented by directed edges. The main reason for interest in the dependency graphs over the standard phrase parse structure is due to the computationally efficient and flexible nature of these graphs and an improved ability to model "non-projectivity" in languages with flexible word orders. An example of a dependency parse for a natural language sentence is shown in Figure 2.2 McDonald and Pereira [2006].

The parse tree representation of natural language text is very similar to the structure shown in Figure 1.1 for analysis of audio content. One can see a clear similarity especially between the phrase parse representation for natural language text and the modeling of event sequences in our paradigm. We will also discuss dependency structures and scenarios in which they would be useful in modeling sound sequences and the concept of non-projectivity and how it relates to our work in our proposed work section in greater detail.

The main difference between the parsing paradigm for natural languages and the discovery of structure among audio events is as follows: the parses for natural language text are syntactic, and enable analyses of the syntactic relationship between words, and of the sentence as a
whole. Relationships between events in audio are semantic, and the accuracies of the inferred dependencies are likely to be more sensitive to the detection of individual events. Further, text used in these settings are usually noise-free, while the analogous discrete sequence for sound (the Acoustic Unit Descriptors) are not likely to be noise-free.

Research in text processing has also seen considerable efforts made toward obtaining semantic parses of text. Shallow semantic parses attempt to identify and label the various arguments for the predicates in the sentence, by assigning labels that enable an understanding of WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW, etc. This has been approached as a tagging task Gildea and Jurafsky [2000] and researchers have applied various machine learning approaches to address it. Further, for the task of analyzing discourse structure instead of individual sentences, efforts have been made to detect semantic relationships between sentences to decide whether one is a paraphrase or contradiction of the other or whether one entails the other. The last has been explored extensively via shared tasks in the Recognizing Textual Entailment (RTE) task in the text Analysis Conference (TAC). Typically, the task of recognizing entailment is treated as a binary classification problem, where the task is to decide whether two sentences are related or not. The two most popular approaches to this task involve using simple bag-of-words classifiers Glickman and Dagan [2005] and using deep analysis of sentence structure Bos and Markert [2005] to decide whether the sentences are related or not.

The basic premise behind the entailment task seems similar to our formulation for sounds, where we can seek to find entailment relationships between various audio segments (or events). Again, however, the two problems are fundamentally different in that the discrete symbol sequences that we have to work with are the sequences of AUDs, which are inherently noisy. Further, while there are various different methods to analyze the content of sentences such as using parse structures, there exist far fewer well-structured information that we can exploit in the case of events (events are sequences of AUDs).

2.3 Semi-Supervised Learning

Traditional classifiers use supervised data with feature and label pairs for each data point for training. However, for a variety of reasons such as human effort and time required for labeling, the cost of obtaining such annotations and the availability of experts capable of labeling the data for the tasks, such labels may not be available in all settings. Unsupervised approaches, on the other hand, make use of large quantities of unlabeled data that is easily available, but do not require annotated labels for the training data. Semi-supervised learning is a special learning setting which makes use of large amounts of unlabeled data in conjunction with a small amount of supervised labeled data to build better classifiers Zhu [2004]. Semi-supervised approaches require less human effort and often provides improved performance making the approach interesting in both theory and practice.

It is worth noting here that semi-supervised learning approaches are not guaranteed to increase performance over unsupervised approaches. For instance, Elworthy [1994] observed that training a Hidden Markov Model Rabiner and Juang [1986] can reduce accuracy under certain initial conditions.

Various algorithms have been developed for semi-supervised learning— self training and co-training are two of the most popular approaches. Self-training uses the labeled data to learn a classifier and then uses this classifier to predict classes on the unlabeled data. The unlabeled data points that this classifier can make predictions on with the most confidence are added to
the labeled training set, and this process is repeated, with the classifier using its own predictions to teach itself. Self-training approaches have been used for word sense disambiguation Yarowsky [1995] and for detecting objects in images Rosenberg et al. [1995].

Co-training Blum and Mitchell [1998] assumes that features can be split into two sets and that each sub-feature set can be used to train a good classifier, provided the two sets are conditionally independent given the class. Initially, two separate classifiers are trained with the labeled data, on the two sub-feature sets respectively. Each classifier then classifies the unlabeled data, and teaches the other classifier with the few unlabeled examples that they can classify with the most confidence added to the supervised data set. Each classifier is now retrained with the additional training examples given by the other classifier, and the process repeats. Thus, the two classifiers must agree on the much larger unlabeled data as well as the labeled data. Nigam and Ghani [2000] showed empirically that co-training performs well if the conditional independence assumption indeed holds.

In the case of sound data, semi-supervised learning approaches are important for 2 reasons. First, for the kind of semantic structure we are attempting to discover from audio, annotated data with annotations providing information about event sequences in the audio as well as relationships between the various events are not going to be easily available. Annotating long audio files with such structured annotations will likely be expensive in terms of human effort required. As such, even if annotations can be obtained, there will likely be only a small amount of supervised data available, necessitating efforts to leverage easily available unsupervised data. Second, audio datasets which provide class labels on the entire files are available. However, especially in the case of user-generated data available easily on platforms such as Youtube, labels simply refer to the subject that the audio was attempting to capture; the audio does contain significant amounts of off-topic content in that some of the portions may be significantly different from the indicators provided by the labels, as voice overs and extraneous commentary or music are overlaid on top of the original audio.

In the second case, one can think of a training audio file as being a bag of data points, where the label indicates that at least one of the data points in the bag belongs to the label. Explicit labels are not provided for the individual data points in the bag. Typically, audio processing systems treat the entire set of data points in the bag as positive instances while training, which is less than ideal. Such setting are known as multiple-instance learning Zhou [2004] in the literature and have been used for various tasks, such as drug activity prediction Dietterich et al. [1997] and visual tracking Babenko et al. [2009].

Multi-instance learning has been applied to the problem of music information retrieval by Mandel and Ellis [2008]. They used tags available from the metadata, which apply to the musical content at various granularities to evaluate the effectiveness of a multi-instance learning approach on this task. While the tags are applicable to the audio file as a whole, some of them are not available to segments of the file, e.g. while the Coltrane track Giant Steps could be tagged saxophone, the tag is not applicable to the piano solo. The work found that the multi-instance approach worked better than the baseline at recovering clips of audio.

The other learning paradigm that is of interest due to the scarcity of labeled data for our tasks is that of active learning Settles [1995]. The main idea behind active learning is that a learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by an oracle (e.g., a human annotator). Active learning is used in a variety of applications, where unlabeled data may be abundant or easily obtained, but labels are
difficult, time-consuming, or expensive to obtain.

A learner in an active learning setting is allowed to pose queries in 3 standard scenarios. The first is known as membership query synthesis Angluin [1988], where the learner may request a label for a datapoint from anywhere in the input space, including instances generated by the learner. The main disadvantage of generating instances automatically is that in cases of labels that require human perception, the generated data may not be meaningful to a human annotator.

An alternative to the membership query synthesis is the concept of stream-based selective sampling Cohn et al. [1990] where unlabeled instances are drawn sequentially, and the learner has to decide whether to query for a label or not. Various approaches have been developed to decide whether or not to request a label, using various measures of informativeness and strategies. A third paradigm takes advantage of the fact that large amounts of unlabeled data are available at once and do not need to be queried sequentially to be obtained. This motivates pool-based sampling Lewis and Catlett [1994], where the most informative examples are drawn from the pool of unlabeled data, which is assumed to be static.

2.4 Similarities to Image Processing Approaches

In the area of image processing, researchers have attempted to employ grammars to improve the performance of computer vision systems. Early efforts to building such systems attempted to exploit the fact that the evolution of a visual scene is guided by different conventions, such as the laws of traffic or social conventions (such as expecting people to sit on chairs, or expecting to see food on tables). Subsequently, research efforts have sought to generate conceptual descriptions from sequences of images Nagel [1988] using intermediate levels of description using verbs to describe temporal changes, events and histories in order to go from frame level understanding to a schematic story.

To the best of our knowledge, Christensen et al. [1996] were the first to suggest using grammars for the task of describing a scene, using a rule-based grammar for this task. Subsequently, the problem of object detection has motivated the formulation of grammar based approaches, where the objects of interest are modeled as being composed of parts which are objects, as well. Such models Felzenszwalb and McAllester [2010] also make efforts to distinguish between different compositions leading to the same object, which would potentially provide semantic information, e.g. distinguishing a smiling face from a frowning face, using the same parts to detect the face. Felzenszwalb et al. [2010] outline a cascade detection algorithm for a general class of models dened by a grammar formalism. This class includes tree-structured structures as well as richer models that can represent each part recursively as a mixture of other parts.

Recently, image processing researchers have proposed the generation of hierarchies from images using both visual and semantic information. Generative models have been proposed for the same Li et al. [2010c], and the success of the models in generating hierarchies using image information augmented with text was measured on image classification tasks as well as by human judgment. While low-level image features, such as pixels, have proved to be strong features for many image tasks such as classification, work by Li et al. [2010b] found that the use of features based on detected objects in an image provide complementary information to the low-level features, and can be used to enhance performance.
Chapter 3

Preliminary Work

In this chapter, we review our preliminary work towards this dissertation. In Chapter 1, we introduced a hierarchical structure that we could use to make inferences at various levels regarding the content of the audio (the structure was illustrated in Fig 1.1). In this dissertation, we investigate techniques that will allow us to automatically learn structured information from weakly supervised or unsupervised data.

The primary hypothesis behind this dissertation is that sound has its own language and structure. In Figure 1.1, we have 3 latent layers over of the observed sound data. The lowest layer among these (the one closest to the observed sound) consist of a sequence of atomic sound units, which we call acoustic unit descriptors (AUDs). In this language for sounds, the AUDs can be considered analogous to the alphabet.

The middle layer in the hierarchical structure is the event layer. Here, the term event refers to semantic events that are expected to span segments of sound, and therefore, sequences of AUDs. We note that such a definition may be specific to certain applications, and that we may require an additional sub-event layer, where sub-events span AUDs, and events span sub-events. The root or highest layer in this hierarchical structure contains information about the semantic class or topic that generated the sequence of events.

We note that such a hierarchical structure may be learned jointly, or layers may be treated individually when inferring the structure. In our preliminary work, we work primarily on the layer containing the AUDs, and validate our hypothesis that AUDs capture semantic information by showing that automatically learnt models for AUDs perform well on audio processing applications.

This chapter is organized as follows. In Section 3.1, we discuss AUDs in greater detail, including the intuition and an unsupervised learning paradigm. In Section 3.2, we present experimental results that demonstrate the efficacy of this approach on real-world data. We conclude the chapter with a discussion in Section 3.3.

3.1 Acoustic Unit Descriptors

In the hierarchical structure introduced in Figure 1.1, the latent layer containing the acoustic unit descriptors is closest to the observed sounds. Our hypothesis in attempting to formulate a language for sound is that sound can be modeled with a small number of lower level units (AUDs, in our case). Individual AUDs do not necessarily capture semantic information, but sequences or patterns of AUDs should capture semantic information. Every instant of an audio file is part of one such unit, and the entire audio stream can be transcribed in terms of these units. If
every AUD were to have distinct semantic identity, the number of AUDs required to represent all audio would be very large\(^1\). Instead, we hypothesize that if we used just a small number of AUDs, the patterns in the transcriptions of audio recordings in terms of these AUDs will still be characteristic of the larger events in the audio. The transcription of audio in terms of AUDs also results in a mapping from the acoustic (or acoustically derived) feature space to a discrete symbol space.

One could think of AUDs as being to sound what phones are to speech. However, because audio (in general) is so diverse and variable, one cannot expect to be able to interpret individual AUDs. Further, since AUDs are a synthetic concept, it is not possible to have supervised transcripts of recordings in terms of AUDs. As a result, the training process for AUDs is fundamentally unsupervised. We discuss the process of parameter learning in greater detail in Section 3.1.1.

Let us illustrate the intuition behind our formulation with an example– consider sounds from a baseball video. Fig. 3.1 shows three video frames from such a recording. The bat makes contact with the ball, producing a sharp metallic sound. This is followed by footsteps running, and finally, cheering as teammates congratulate the player. A listener familiar with baseball would be able to infer from the sequence of sounds that they may be from a baseball game, and that a hit or a run may have occurred. Although the precise sounds produced and their sequences may vary in nature, the overall pattern of sounds is still characteristic of the event.

![Figure 3.1: Example of a sequence from a baseball video.](image)

The sound of the ball being hit, footsteps, cheering, etc. are all atomic sound events that characterize the larger event of the run being batted in. Moreover, besides these key events, there are other individual nondescript atomic events such as rustling, silence, etc. which occur in the recording. In fact, every instant of the audio may be considered to be a part of one such atomic event. The overall pattern of occurrence of these atomic events characterizes the larger event, and our work tries to mine AUD sequences to be able to both identify the larger event as well as attempt to find structured sub-events that lead to the conclusion, if any, as shown in Fig 1.1.

[ We briefly digress here to point out an instance where one layer for events (instead of an event and a sub-event layer) could be expected to be enough. In audio of a baseball game, the sub-events that constitute events (e.g. plays) are short-duration events— bat hitting a ball, throwing or catching a ball– and most of these could be expected to be captured within individual AUDs. In case of events that are more complex and of significantly longer duration, such as a car chase in a movie, it may be necessary to have a sub-event layer to model the events better. ]

\(^1\)Nonetheless, we do expect that individual AUDs do capture some underlying semantics, even though we may not be able to quantify this in terms of generally understood semantic concepts
The process of learning the parameters for the AUDs is described in Section 3.1.1. We propose a maximum-likelihood solution that jointly estimates the parameters of the HMMs for the AUDs and the (potentially, category-specific) language models of audio from a training corpus. The solution is analogous to the unsupervised and semisupervised automatic learning of sub-word units in speech Bacchiani [1999], Singh et al. [2002].

3.1.1 Learning AUD Parameters

The first problem that we encounter in this formulation is that of representing the data with a sequence of AUDs that are likely to have generated the data. The concept of AUDs is similar in principle to that of acoustic segment models (ASM, henceforth) Lee et al. [1988]. While the ASM was derived for speech, AUDs make no assumptions about the nature of sound in general. The ASM also used an acoustically derived lexicon for word models based on subword segment models, while there is no equivalent available for AUDs which are learnt unsupervised. Further, AUDs do not spot specific events in the audio stream – the entire audio stream can be transcribed in terms of these units, i.e. every segment of audio is part of some AUD.

The problems we must address now at training time are twofold: A) Learning of AUDs: Given a set of training audio recordings, we must learn the set of AUDs, B) Learning of AUD distributions: We need to learn statistical characterizations of the patterns of AUD sequences for audio from different categories. The learning process is inherently unsupervised, since the AUDs are a synthetic concept, and one cannot obtain ground truth transcriptions of audio in terms of AUDs.

Each individual acoustic unit can be modeled using any structured model– one instance would be using a Hidden Markov Model formalism, as described in our prior work Chaudhuri et al. [2011a]. We represent the audio signal as a sequence of mel-frequency cepstral vectors, as is the norm in speech recognizers. We model each AUD by a Bakis-topology HMM with Gaussian-mixture state-output densities. Since we are primarily interested in characterizing the AUDs, rather than interpreting their semantics, learning the AUDs is equivalent to learning the parameters of the HMMs for the AUDs. We model the distribution of the AUD sequences as N-gram language models over the vocabulary of AUDs. One can choose to model the N-gram sequences in a class-specific manner where given class labels for the recordings in the training data, separate language models are learnt for each class. In our treatment of the learning process below, we will discuss the case where class-specific language models are trained, since the case where there is one class-independent model is a special case of this. Note that the AUD models are shared by all classes.

We cast the learning problem as one of maximum likelihood estimation. We are given a collection of audio recordings $\mathcal{D}$. Assigned to each recording $D_i$ in $\mathcal{D}$ is a class label $C_i \in \mathcal{C}$, where $\mathcal{C}$ represents the set of all classes. Although not necessary, we will assume that each recording is entirely assigned to only one class. Each audio recording $D_i$ has an unknown transcription $T_i$ as a sequence of AUDs. The AUDs are modelled by HMMs, whose parameters we collectively represent as $\Lambda$. The transcriptions of all recordings belonging to a class $C$ are assumed drawn from an $N$-gram language model $H(C)$. The HMM parameters $\Lambda$ and the set of language models for all classes $\mathcal{H} = \{H(C) \forall C\}$ are unknown and must be estimated from the data. We assume that the total number of AUDs $K$ is known. In reality, $K$ is a hyperparameter that may be optimized.

We assume the dependencies shown by the graphical model in Fig 3.2: the acoustic realization of any recording depends on its transcription and not directly on the language model for the class.
So also, the transcriptions only govern the acoustic realization and do not directly relate to HMM parameters.

![Graphical model for each data point D. Circles represent random variables and rectangles represent parameters.](image)

**Figure 3.2:** Graphical model for each data point D. Circles represent random variables and rectangles represent parameters.

The maximum likelihood estimate for $\Lambda$ and $H$ is given by:

$$\Lambda^*, H^* = \arg\max_{\Lambda, H} P(D|C(D); \Lambda, H) \quad (3.1)$$

Here $C(D)$ represents the classes assigned to each $D$ in $D$. In the notation above terms to the right of the semicolon are parameters, while remaining terms are random variables.

In principle, the above estimator must consider all possible transcriptions for any $D$. Instead we will approximate it by only considering the most likely transcription for any $D$. Also, assuming that individual recordings $D$ are independent, and that the class is represented primarily through the language model for the class, the estimator changes to:

$$\arg\max_{\Lambda, H} \prod_{C_i} \max_T P(D_i, T; \Lambda, H(C)) \quad (3.2)$$

We obtain the above estimate using the iterative algorithm in Algorithm 1. In the algorithm, superscripts appearing against the parameters indicate the iteration in which the estimate for the parameter was obtained.

**Algorithm 1** Iterative algorithm for learning AUDs and LMs

\[
\begin{align*}
T_i^{r+1} &= \arg\max_T P(T|D_i; H(C_i)^r; \Lambda^r) \quad (3.3) \\
\lambda^{r+1} &= \arg\max_\Lambda \prod_{D_i} P(D_i|T_i^{r+1}; \Lambda) \quad (3.4) \\
H(C)^{r+1} &= \arg\max_H \prod_{D_i, C_i=C} P(T_i^{r+1}; H) \quad (3.5)
\end{align*}
\]

It is simple to show that Algorithm 1 is a hill-climbing procedure that results in ever increasing likelihood for the data: Equation 3.3 ensures that

$$P(D_i, T_i^{r+1}; \Lambda^r, H(C)^r) \geq P(D_i, T_i^r; \Lambda^r, H(C)^r) \quad (3.6)$$

and Equations 3.4 and 3.5 ensure that

$$P(D_i, T_i^{r+1}; \Lambda_i^{r+1}, H(C_i)^{r+1}) \geq P(D_i, T_i^{r+1}; \Lambda_i^r, H(C_i)^r) \quad (3.7)$$

Equation 3.3 simply represents the automatic recognition of $D_i$ using HMMs with parameters $\Lambda^r$ and can be performed with the Viterbi decoder of a speech recognizer. Equation 3.4 is the
learning procedure for HMM parameters $\Lambda_i$ given the recordings $D_i$ and their transcriptions $T_{i}^{r+1}$, and can be performed using the Baum-Welch training module of any recognizer. Equation 3.5 represents the procedure for learning an $N$-gram language model $H(C)$ from the set of all transcriptions $T_{i}^{r+1}$ of all recordings belonging to class $C$.

Algorithm 1, however, requires an initial transcription for all recordings. We obtain this by segmenting all recordings by merging adjacent analysis frames, and finally clustering the obtained segments into $K$ clusters. The sequence of cluster identities corresponding to the segments composing any recording form the initial transcription for that recording.

The mapping between the AUDs and the observed acoustic data (we work with MFCC features derived from the raw audio) is stochastic. Given models for these AUDs, we can decode new audio in terms of the AUDs, and use $n$-best lists to obtain discrete sequences from the audio. However, we recognize that such decodes are likely to be noisy.

The iterations of Algorithm 1 lead to progressively improved joint AUD and language models. As we shall describe in Section 3.2, we experimented with both a class-specific language model setting as well as a class-independent language model setting, on different applications. The experimental results we describe also show that our models of the AUDs do, in fact, capture some underlying semantics.

3.2 Applications of AUDs to Audio Processing Tasks

In this section, we describe applying an AUD-based technique to 3 common audio processing tasks. First, to make the challenges of the tasks clearer, we start with a description of the datasets we use in our experiments. We then describe our experiments and results on the task of audio classification in Section 3.2.2, on the task of audio retrieval in Section 3.2.3 and in an event detection-like setting in Section 3.2.4.

3.2.1 Datasets

In the completed preliminary work in this proposal, we work primarily with 2 datasets, that have very similar characteristics. These datasets are from the Multimedia Event Detection (MED) task of TREC Vid, and are provided by NIST. The data belong to one of several semantically labeled classes, and class labels for each recording is provided in the dataset. We performed our experiments on data from the 2010 MED [2010] as well as 2011 MED [2011] versions for this task. In our work in this dissertation, we only use the audio in the recordings, since our objective is to evaluate our ability to characterize the audio. While these characterizations could potentially be combined with the video, we have not attempted to do so yet. We do not use any annotations besides the basic activity labels provided, and use no other external data of any kind, except to set up baseline systems for comparison.

The TREC Vid 2010 Multimedia Event Detection dataset comprises 1746 total clips of training data, totaling about 56 hours in length, and the 1724 clips of test data about 59 hours long. The recordings are publicly available, user-generated multimedia content uploaded to internet hosts. Each video is annotated with one of 4 labels – making a cake, batting in run, assembling shelter and other, identifying the kind of activity being performed in it. The class other appears to be a catch-all class consisting of all videos that do not belong to the first 3 classes. Participants in the NIST MED evaluation were required to retrieve recordings from the testset that included a queried activity or event, and largely focused on the video features available. The use of the
audio features was usually limited to speech transcriptions Li et al. [2010a], and detection of pre-specified sound types in the audio Hill et al. [2010].

The training data for the MED, 2011 task consists of 5 events—*attempting a board trick, feeding an animal, landing a fish, wedding ceremony, working on a woodworking project*, which are referred to as E001-E005 respectively. The data contain a number of files that do not belong to any of these classes. For each of the files in the test data, we know which of the 5 events it belongs to or if it does not belong to any of them. The training data contains about 2.4K files (with 815 belonging to the 5 events of interest) while the test set consists of about 10K files (with 496 belonging to the 5 events of interest).

### 3.2.2 Multi-Class Audio Classification

Earlier in this chapter in Section 3.1.1, we discussed a learning process for the AUDs and their distributions. In the experiments we describe in this section, we work on the multi-class data in the MED, 2010 task and use class-specific language models. The models for the AUDs are shared by all these classes.

Given the HMM parameters \( \Lambda \) for all AUDs, and the set of language models \( H(C) \) for all classes \( C \in \mathcal{C} \), we can now classify a new audio recording \( D \) into one of the classes by Bayesian classification:

\[
C^* = \arg\max_C P(D|C; H(C), \Lambda) P(C)
\]

To compute \( P(D|C) \) we must sum over all possible transcriptions of \( D \), which is generally computationally intractable. Instead, we can employ the common approximation of only considering the most likely transcription:

\[
\arg\max_C \log P(C) \max_T \log P(D, |T ; \Lambda) + \log P(T; H(C))
\]

(3.8)

Here \( \max_T \log P(D, |T ; \Lambda) + \log P(T; H(C)) \) is the log likelihood of the most likely transcription of \( D \) for class \( C \) for and can be computed by the decoder of a speech recognizer. \( \log P(D, |T ; \Lambda) \) is the acoustic score \( A(D, C) \) for class \( C \) and \( \log P(T; H(C)) \) is the language score \( L(D, C) \) for the class.

However, we do not use the above procedure directly for classifying the recordings. Instead we employ a second-level classifier that uses the acoustic and language scores for the class as features. The primary reason for doing this is that even though the decoding uses a scale factor for the language model with respect to the acoustic model, there is no notion of discriminating between acoustic and language model scores with different classes. The weights learnt in the second stage of the process enable comparison of the total scores between classes.

Let \( F(D, C) = [A(D, C) L(D, C)]^\top \) be a feature vector representing the acoustic and language scores for the data \( D \) computed for class \( C \). For each class \( C \) we define a two-dimensional weights vector \( W_C \). Classification is performed using the following classification rule:

\[
C^* = C : W_C . F(D, C) > W_{C'} . F(D, C') \forall C \neq C'
\]

(3.9)

To train the classifier we learn weights \( W_C \) for each class as follows: For each training instance \( D \) belonging to class \( C_D \) we decode \( D \) using \( H(C) \) to obtain \( F(D, C) \) for every class \( C \). A training instance is correctly classified if:

\[
W_{C_D} . F(D, C_D) > W_C . F(D, C) \forall C \neq C_D
\]

(3.10)
Algorithm 2 Learning weights for each class

\[ M = \text{maxiter}; \ i = 0; \ v = 0 \]
\[ w_c = (0, 0), \forall c \in C \]
\[ w^{(0)} = \{w_1, w_2, ..., w_{|C|}\} \]
for \( m = 1 \) to \( M \) do
  for \( j = 1 \) to \( \tau \) do
    \[ w^{(i+1)} = \min_v ||w - w^{(i)}|| \]
    s.t. \( S(x_j, y_j) \geq S(x_j, y_c), \forall y_c \)
    \[ v = v + w^{(i+1)} \]
    \[ i = i + 1 \]
\[ w = v/(N \times \tau) \]

The weights \( W_C \) can be learned to maximize classification accuracy on the training data.

We optimize an objective function with an iterative algorithm described in Algorithm 2. The algorithm is an online margin learning algorithm— it seeks to update the vector of weights after it encounters each new instance so that the weighted score using the feature vector generated by the model for the true class label is greater than the weighted score using models for the other class labels. Let us assume that for each training instance \((D)\), the features scores (features) have been computed, and the label \((y_j)\) is known, and there are \( \tau \) such training instances. Instead of simply requiring that the score using the true model be greater than the score using the other labels, one can modify this formulation to include a margin by which the score of the true label should be greater than the score of other labels. This can be done by adding a positive term to the right hand side of the constraint. This term may be a constant or may be the result of some loss function.

Experimental Results

We learned models using the data for the 4 classes in the MED10 training data, as described earlier. The 4-class classification task contains an extremely skewed majority class: the other class has far more recordings than the other classes. In order to experiment with a more balanced dataset, we also experimented with 3 class classification, leaving out the data from the other class.

Table 3.1 reports accuracy (recall) for the 3 best settings obtained by classifying data directly using their Viterbi decode scores. We also compare with a simple baseline model, where we simply model each of the four classes with an ergodic HMM and perform Bayesian classification. The HMMs and the a priori class probabilities employed were tuned for best performance in the last case. Table 3.2 reports results using weighted MIRA classifier.

Overall, our experiments indicate that bigram language models outperform both unigram and trigram models on this task. Further, using 64 sound units appears to outperform systems that use more sound units on the 3 class classification task, but it doesn’t do as well on the 4-class task. This supports the intuition that more units better capture a larger set of sounds. The MIRA classifier is generally significantly superior to classification based on Viterbi scores alone.

Employing our approach on the MED dataset involves significant challenges. For instance, the other class is not consistent in content, and contains a wide array of different audio and video. Besides the other class, the remaining 3 classes are not all well-structured. Events in the batting_in_run class have audio structure to them, as discussed earlier, but the audio in the
Table 3.1: Classification based on Viterbi decoding scores

<table>
<thead>
<tr>
<th>System</th>
<th>3-class</th>
<th>4-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 symbols 2-gram</td>
<td>76.51%</td>
<td>64.79%</td>
</tr>
<tr>
<td>64 symbols 3-gram</td>
<td>75.00%</td>
<td>53.10%</td>
</tr>
<tr>
<td>200 symbols 2-gram</td>
<td>41.88%</td>
<td>67.42%</td>
</tr>
<tr>
<td>Class-specific HMM</td>
<td>36.88%</td>
<td>43.56%</td>
</tr>
<tr>
<td>Random</td>
<td>33.33%</td>
<td>25%</td>
</tr>
</tbody>
</table>

Table 3.2: Classification using MIRA classifier

<table>
<thead>
<tr>
<th>System</th>
<th>3-class</th>
<th>4-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 symbols 2-gram</td>
<td>81.61%</td>
<td>73.61%</td>
</tr>
<tr>
<td>64 symbols 3-gram</td>
<td>80.30%</td>
<td>59.72%</td>
</tr>
<tr>
<td>200 symbols 2-gram</td>
<td>55.63%</td>
<td>77.08%</td>
</tr>
</tbody>
</table>

assembling_shelter and making_cake classes are widely varied. Table 3.3 compares the accuracy for each class for the 200 symbol bigram models with the simple baseline class-specific HMM models on 4 class data.

Table 3.3: Category specific accuracy for the various classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Class-specific HMM</th>
<th>200 symbol 2-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>assembling_shelter</td>
<td>31.11%</td>
<td>44.00%</td>
</tr>
<tr>
<td>batting_in_run</td>
<td>34.62%</td>
<td>59.62%</td>
</tr>
<tr>
<td>making_cake</td>
<td>43.86 %</td>
<td>24.14%</td>
</tr>
<tr>
<td>other</td>
<td>64.67%</td>
<td>94.70%</td>
</tr>
</tbody>
</table>

It is not clear to us why the making cake class is better predicted with class-specific HMMs, but we believe it may have something to with the fact that audio corresponding to making a cake class appears to contain speech only in most cases and non-speech sounds do not have any class-specific interpretation, or do not occur sufficiently close for the sequence information to be used by our model. Our system does a very good job of identifying the other class, which is useful in reducing the number of false positives for the other classes, and makes this approach higher precision than the baseline.

Qualitatively, on analyzing the transcriptions generated on the training data by the iterative learning procedure, we find that it does a consistent job in identifying some sounds, such as the sound of a baseball bat hitting the ball or clapping, but runs into trouble when encountering other sounds, such as speech. Speech information appears to be distributed among various units. A couple of things could be done to improve this in future work- first, identifying speech segments as a step before sound unit learning, possibly in a supervised manner to help the system focus on non-speech acoustic events, or use a speech detector to ensure that speech events are constrained within a few sound units; second, we could start with a small amount of supervised data, that specify class-specific characteristic sounds to help the system converge to a better solution instead of building a sound dictionary from scratch.

In conclusion, we would like to note that the method to automatically learn sound units is a
potent one. In the context of audio data, it is perhaps necessary to add in a layer of supervision in order to help add semantic information. Detection and recognition of speech and using the transcripts to help distinguish between categories should produce significant improvements in performance.

3.2.3 Audio Retrieval

The audio retrieval task is a modification over the multi-class classification task, described in Section 3.2.2. In this work Chaudhuri et al. [2012a], we attempt to detect all recordings of a specific event type in a large collection of youtube-style recordings—thus, for each event type, we have a binary one-against-all setting. Since the events in the task are semantically defined, one might expect that such a task might be addressed by looking for semantically meaningful acoustic phenomena that correlate well with the events in the audio. Instead, we attempt to perform detection using AUDs which have no clear semantic interpretation. The process of using AUD sequences to describe the audio is done according to Equation 3.3, given the models for the AUDs and their distributions. Here, to characterize the recordings in terms of AUDs, we simply use unigram counts. While \( n \)-gram patterns clearly carry information, they may be difficult to detect because the decoding of audio into AUDs is inherently noisy. Further, \( n \)-gram patterns over the larger universe of sounds would be harder still because of the larger set of possible \( n \)-grams.

Given training data with event labels for audio files, we use a classifier that uses features extracted over the AUD transcriptions for the audio files to decide whether a test file belongs to a specific event type or not. Our experiments show that the AUDs perform very well at the task of detecting the events. In fact, retrieval based on AUDs is superior to that obtained using semantically meaningful units—phoneme models learnt from speech data, and models for sound learnt from a library of common sounds. In this section, we first describe the AUD-based feature sets we use in Section ???. In order to set up a baseline for comparison, we then describe a feature set obtained from the audio using semantically motivated units to transcribe the audio in Section 3.2.3. We then describe the classifier used for this task in Section 3.2.3, and finally, we discuss our experimental results in Section 3.2.3.

Features for Audio Retrieval

The process of designing a feature set is often critical to classification tasks. Different feature sets using the same learning framework often yield different results because one set of features may be more suited to the task, or might capture more discriminative information than another. Temporal behavior of features has been shown to be important for audio and music classification, and the use of models of auditory perception in feature sets have proven to be better than MFCC based feature sets for the same tasks McKinney and Breebaart [2003]. Different feature sets have been shown to affect performance in other applications as well, such as web document classification Qi and Davison [2009].

In this section, we report results from experimenting with feature sets derived from AUD sequences in the data. AUDs can be used to transcribe audio as a sequence of discrete symbols. Even though the process of learning the AUDs does not use any discriminative class-specific information, we expect that different AUDs will capture different kinds of acoustic phenomena. At the same time, different classes of audio data will contain different distributions of the various acoustic phenomena, resulting in different distributions of AUDs. We extract the frequency of
occurrence of each AUD in the transcription for an audio file. The \{AUD, AUDfrequency\} pairs thus obtained are used to construct a feature vector for each data point.

In order to analyze the AUD-based features, we also define two other feature classes— a binary feature vector that indicates occurrence of each AUD based on the transcription, and a feature vector where the feature value for an AUD is the total number of frames spanned by all instances of the AUD in the transcript for the audio.

We model the data with a total of 64 AUDs, with each AUD modeled as a 5-state HMM, with a mixture of 8 gaussians used to control the emissions and a unigram model over the AUDs to control inter-AUD transitions. The unigram distribution over AUDs is learnt from the entire data at training time, and not in a class-specific manner. The audio is represented as a sequence of mel-frequency cepstral (MFCC) vectors augmented with deltas and double deltas.

**Baseline using Semantically Meaningful Features**

It seems intuitive that using semantically meaningful units would be a reasonable way of approaching the task of detecting semantically defined audio event classes. We set up 2 other models for learning acoustic units in a semantically meaningful manner to compare with the AUD based features.

The first is based on phoneme models obtained from speech data. As mentioned earlier, AUDs for sound can be thought of as analogous to phones for speech. Further, in the prior work described in Chaudhuri et al. [2011a], we found that AUD-based models did not do a very good job of identifying audio that contained speech \(\text{e.g. making cake}\) class in the MED10 dataset, perhaps because the limited number of AUDs in the vocabulary resulted in them modeling other (non-speech) sounds more closely. As a result, we decided to explore explicit speech models to see if they could augment AUD models in improving performance.

The HUB4 corpus Pallett et al. [1996], consisting of business broadcast news audio, was used to train 40 phoneme models (as well as 5 fillers), using 40 filters between 50Hz to 6800Hz. Each phoneme was modeled by a 3-state HMM, with the emissions being governed by mixtures of 16 gaussians.

Similar to the manner in which we generated AUD sequences, we can use the phoneme models in an all-phone decode setting to obtain a transcription of any audio file as a sequence of phones. We can generate a count vector of these phones for every training file, resulting in a 45-dimensional feature vector for each file.

The second feature set is based on an audio library, which consists of a number of common sounds. Audio libraries have been used for various retrieval tasks Slaney [2002b] using human annotations to build detectors for the sounds. In this paper, we use the Art of Foley Sound Effects Library Fol [2005], which consists of 480 individual audio events. We trained HMM models for each of these events with 5 states and emissions governed by a mixture of 8 gaussians. MFCC features, used to represent the audio were generated with 32 filters between 100Hz and 5200Hz. As before, we learn HMM models for the various sounds to obtain a transcription of an audio file as a sequence of sounds. Count vectors are generated for the various sounds, resulting in a 480 dimensional feature vector.

**Random Forest Classifier**

The task we tackle requires us to detect data as belonging to a particular class or not. We would like to be able to search for and retrieve all files belonging to a particular class. Hence, we train
binary classifiers for each audio class to detect whether a test file belongs to the class or not (one versus all).

The experiments we report in this section employ a random forest Breiman [2001] based classifier. Random forests are an extension of decision tree classification techniques, where the training process grows many trees instead of a single one. Given a new test file, each of the trees in the forest returns a class label, which is used in a weighted vote to determine the final predicted label.

While we could have used any classifier for this task, we chose random forest classifiers as they are resistant to overfitting. Training random forests is done by sampling with replacement from the training data, with about one-third of the training data typically held out. This held out data is used to get an estimate of the error as trees are added to the forest. The trees in the forest are grown as far as possible, and pruning is not used. For details of the training process, the reader is referred to Breiman [2001].

Experimental Results

The task we use to evaluate performance is one of detecting all recordings of a specific event type from the MED11 dataset. The training and test data consist of positive instances from the set of 5 classes, and a number of files that do not belong to any of those classes. For each class \( i \), where \( i \in \{1, 2, \ldots, 5\} \), the positive instances for that class are taken along with all the other files (which are negative instances for that class), and a detector is trained for that class.

At test time, for each file in the test set, the detector for class \( i \) predicts whether or not the test file belongs to the class \( i \). We evaluate performance using Missed Detection (MD) and False Alarm (FA) rates, which are defined as follows: suppose there are \( N_t \) test files, of which \( N_i \) are labeled as belonging to class \( i \) by the detector for the class. However, the test set contained \( C_i \) files belonging to class \( i \), and \( D_i \) of these were correctly detected. Then:

\[
MD = \frac{C_i - D_i}{C_i} \\
FA = \frac{N_i - D_i}{N_t - C_i}
\]  

We evaluate performance using the area under ROC curves which measure MD rate against FA rate. Since both missed detections and false alarms are measures of error, the lower the area under the curve, the better it is.

We evaluate the result of the various feature sets on performance, with ROC curves based on missed detections and false alarm rates. We described the various feature sets in Section 3.2.3 and 3.2.3 the various feature sets—binary indicators for AUD presence (AUDs_binary, henceforth), AUD count vectors (AUDs_count, henceforth), AUD frame vector counts (AUDs_framecount, henceforth), phoneme count vectors (Phone, henceforth) and audio library sound-type count vectors (Foley, henceforth).

In order to obtain the ROC curve: we apply a threshold to the score returned by the classifier to decide whether each datapoint is a positive or negative. Based on the classifier’s prediction over the entire test set, we compute the MD and FA rates for a given threshold. Varying this threshold allows us to obtain the ROC curve.

Fig 3.3 shows the performance in terms of Area Under Curve (AUC) when using the different feature sets for the 5 classes of data (note that lower AUC is better). We observe the following:
Figure 3.3: AUC for the various feature sets (lower is better)

- The best performance overall is obtained in the AUDs_count setting. The AUDs_count-based classifier significantly outperforms both Phone and Foley-based classifiers.
- The AUDs_framecount classifier does not outperform the AUDs_count classifier.
- Interestingly, the AUDs_binary classifier performs reasonably well on this task, and is comparable overall to the Foley.

Figures 3.4, 3.5, 3.6 show the ROC plots for the 5 classes in the MED11 dataset. The red circle in the figures represent the operating point of 75% missed detection at 6% false alarm rate. As shown in Fig 3.3, there is very little difference between the AUDs_framecount and AUDs_count cases.

We note from the results above that a coarse representation using data-driven AUDs with no guaranteed associated semantics outperform semantic units significantly. A limited set of semantically-defined acoustic units such as those in the Foley set or the phoneme list cannot adequately describe all events that occur in even a small collection of audio recordings. In order to cover even a moderate fraction of all possible phenomena that can occur, one would require an impossibly large vocabulary of semantic units. Even if such a set of units were available, confusions in detecting them automatically in a recording could render them ineffective. Our hypothesis is that this is where data-driven units that can learn a vocabulary to fit the data
without making significant prior assumptions can provide a distinct advantage, and this is borne out by our experiments.

Nevertheless, the AUDs obtained through data-driven discovery are not entirely lacking in semantic association. As we see from the AUDs binary setting in the results in Fig 3.3, merely detecting the presence of AUDs in a recording is sufficient to identify the event type with a probability that is significantly better than random. This may be interpreted as an indication that the AUDs do carry characteristic information that can distinguish one event type from another, which may in turn imply that they do capture some underlying semantic. Moreover, the AUDs themselves are generatively learnt without any explicit requirement to be discriminative; yet they perform well on a discriminative task, further strengthening the notion that they capture some underlying semantic in the data.

While we used AUDs to perform retrieval of entire audio recordings, one could also envision using them to detect smaller sub-events within audio recordings. Similar approaches could be employed for segmentation as well as co-occurrence analysis of audio events.

3.2.4 Audio Event Detection-based Applications

In this section, we present our approach to an event detection framework in a supervised setting. Here, the term event detection refers to detecting an event or segment of interest within a larger audio file by identifying the start and end boundaries of the event in the audio stream. As an example, consider recording from a sports game— a significant portion of the file contains non-sporting sequences, such as crowd shots, or commercial breaks. Automatically discovering the segments of interest would provide enhanced navigability to the user looking at the recording. Such a system would also be useful in application pertaining to retrieval of segments pertaining to the query instead of the entire file, automatic generation of highlights, etc.

We describe a large-margin, discriminative training method that uses supervised training data to learn the importance of various features in capturing the context. We use a novel feature set for audio, based on acoustic unit descriptors, which are used to describe the sequence of events in the audio. The AUDs are learnt from the data in an unsupervised manner, as described earlier, although the approach used here is general and one could use sound dictionaries to obtain the sequence of acoustic units Kim et al. [2010]. We use baseball data for our experiments from the MED10 batting in run class, seeking to create a system that would take user-generated content, and remove all non-baseball-action segments from the data. While our experiments in this paper focus on baseball data, the approach is generic and can be applied to any dataset with relevance labels.

This section is organized as follows: we first describe the data used in the experiments in Section 3.2.4. We then describe the learning framework, including the problem formulation, the feature set, and the training paradigm, in Section 3.2.4. Finally, we discuss our experimental results in Section 3.2.4.

Data

We work on a dataset that is focussed on a specific topic— baseball videos. We use the batting in run data from the TREC Vid 2010 Multimedia Event Detection dataset (MED, henceforth) ?.

The main reason for using the data in this class is that it is the most structured and deciding on segments of the audio relevant to the topic batting in run was easiest for this class. For each file, an annotator marked some segments in the audio as relevant. A section is marked as relevant if
it contains baseball action (baseball action refers to sporting action on the field, as distinct from any similar action in the crowd), while other portions of the video not related to baseball action are marked as not relevant.

The training data consist of 54 videos and the test data consist of 52 videos. For both sets of data, the audio is extracted from the mp4 video, and down-sampled to 16KHz, single-channel. The annotator was allowed to use the audio as well as the video to decide which segments were relevant, since the audio doesn’t always make it clear whether the relevant section of the video is over. In the context of the audio for the batting in run data, examples of sections that are not relevant include cutting to the crowd, or having conversations with a friend, or voiced-over segments of audio. For each audio file in the training set, we have a set of segments that are marked as relevant, and the remaining segments marked as not-relevant. We use this data to extract patterns to help us identify segments as relevant or non-relevant.

**Learning Framework**

We showed earlier, in Section 3.1.1, how one can transcribe audio as a sequence of AUDs using Equation 3.3. Further, since we have annotated data that mark segments as relevant, we can use this information to obtain the sequence of AUDs that appeared in the relevant sections of the audio. Thus, for every file in the training data, we have a full transcription in terms of the AUDs, as well as a truncated version, using only the AUDs that appear in the relevant segments. Let us refer to the original, uncompressed transcription in terms of AUDs as $X$ (where $X = x_1x_2...x_m$), and the truncated version using the relevant segments as $Y$ (where $Y = y_1y_2...y_k$, and $k \leq m$).

Thus, if we have $n$ training audio signals, we use the $n$ pairs $\{X_i,Y_i\}$, $\forall i = 1,2,3...n$ as the training data. Thus, we need to train a system using $\{X_i,Y_i\}$, such that at test time, given a transcription of the test audio file in terms of the AUDs ($X_{test}$), the system can generate a compressed version of the AUDs ($Y_{test}$). In order to generate the audio (or video, depending on the application) corresponding to the relevant sections, we can simply synthesize the frames that correspond to the AUDs that were retained.

We would like to note here that the problem setup that we have now is analogous to the problem of text compression by deleting words from text. The problem of text compression has usually been reduced to one of sentence compression and has been approached in a number of ways, including noisy channel models Knight and Marcu [2000], integer programming Clarke and Lapata [2006] and large margin approaches McDonald [2006].

We model our approach to this task as similar to the large margin supervised approaches used in sentence compression research. Unlike the text-based compression approaches which use deep syntactic features based on parse trees generated over the text, we have access only to surface features— AUD identity markers in this case, analogous to the words in the sentences in sentence compression research. We describe the feature set used in Section 3.2.4, and the learning algorithm for training parameters in Section 3.2.4.

**Feature set**

We jointly extract features over the original and compressed transcription pairs in the training data. First, for every pair of consecutive AUDs in the compressed transcription, we consider the bigram feature $y_{j-1}y_j$. We then extract the context information for each of the consecutive AUDs individually in the transcription– both bigram context and trigram context. We also add an indicator feature that says whether or not any 2 consecutive AUDs in the compression were
also consecutive in the original sentence. These features are intended to understand which AUDs are more likely to be relevant to the topic of the audio, and to understand the contexts in which they are retained.

We also add features for AUDs that were dropped from the original, uncompressed transcription of AUDs. This is done in the following manner: for every pair of AUDs that appears in the compressed transcription, we add features corresponding to the identity of the AUDs dropped from the uncompressed transcription. Further, for every AUD dropped from the original transcription, we add a feature that identifies the AUDs nearest to it on either side that were retained; e.g. if the sequence \ldots a_4a_7a_8a_2a_4 \ldots becomes \ldots a_4a_2a_4 \text{ in the compressed transcription, we add features to indicate that } a_7 \text{ and } a_8 \text{ were dropped and that they were dropped so that } a_4 \text{ and } a_2 \text{ appeared consecutively in the compressed transcription. We then add the bigram and trigram contexts of the dropped AUDs as features. The motivation behind these features is to understand the different contexts that indicate whether or not an AUD should be deleted.}

Training

The training process is a large margin online learning algorithm that involves a decoding algorithm that can search the space of all possible compressions in an efficient manner. Given an uncompressed transcription of AUDs $X = x_1x_2\ldots x_m$, we can see that we can generate an exponential number of compressions, depending on the location and number of AUDs dropped.

Thus, for any compression $y$ of an uncompressed transcription $X$, we need to score the compression such that the selected compression $y^* = \arg \max_y \text{Score}(X, y)$. In order to be able to efficiently compute scores over the entire space of compressions, we need to factor the function that computes the score. We do this as follows—suppose the compression $y$ contains $k$ AUDs. Then:

$$\text{Score}(X, y) = \sum_{j=2}^{k} g(X, y_{j-1}, y_j) \quad (3.13)$$

The function $g(X, y_{j-1}, y_j)$ represents a weighted scoring function that extracts features on pairs of AUDs that appear consecutively in the compression (as described in Section ??) and the score is obtained as a weighted sum of the feature values:

$$g(X, y_{j-1}, y_j) = w \cdot f(X, y_{j-1}, y_j) \quad (3.14)$$

Thus, we can compute the score between every pair of AUDs that could possibly be present consecutively in the compressed transcriptions. Note that this set of AUDs is the same as every pair of AUDs in the uncompressed transcription $x_p, x_q$, such that $p < q$. Thus, we compute the function $g$, for every pair $x_p, x_q$, such that $p < q$, in the original uncompressed transcription of AUDs.

Now, we can compute the best compression for the uncompressed AUD sequence $X$ using dynamic programming over the factored scores. Before we do so, however, we need to modify $X$ slightly, so that we have a dummy start and end position at the beginning and end respectively. We can continue to assume without loss of generality that the new $X$, with a START symbol at the beginning and an END position at the end is of length $m$. (At training time, the START and END symbols are inserted to the compressed AUD sequences as well, so that they are never
dropped.) We define our dynamic programming table as follows:

\[
T[1] = 0.0 \\
T[i] = \max_{j<i} T[j] + g(X, x_j, x_i), \forall i > 1
\]

(3.15) (3.16)

The table \( T \) contains entries from \( i = 1, 2, 3, \ldots, m \) when the uncompressed sequence of AUDs \( X \) is of length \( m \). The entry at position \( i \) in the table, \( T[i] \), is the score of the compression that ends at position \( i \). Thus, the score of the best scoring path through the AUD sequence is given by \( T[m] = T[END\_symbol] \) given the way we factored the scoring, and the path can be found by keeping backpointers to remember where the best scoring path at any index came from.

The process described above is the one used to generate compressions at test time, when we have an input uncompressed sequence of AUDs. However, it is also used at training time, as we shall now describe. We would like to obtain the set of parameters \( w \), such that we can generate the compression \( y \) for the training instances \( X \) with as little error as possible.

We use a discriminative large margin online learning algorithm called Margin Infused Relaxation Algorithm or MIRA Crammer and Singer [2003] to train weights. The algorithm is summarized below in Algorithm 3. In the algorithm, \( Y_h \) refers to the hypothesized compression for the training example \( X_t \), using the weights from the previous iteration. \( S(X_t, Y_t) \) refers to the score for the true compression \( Y_t \) in the training data. \( L(Y_t, Y_h) \) represents a loss function that represents the margin in this algorithm. The training data is represented as a set of pairs \( \{(X_j, Y_j)\}_{i=1}^{N} \).

**Algorithm 3** Learning weights using MIRA

\[
R = \text{maxiter}; \; i = 0; \; v = 0; \; w_0 = 0 \\
\text{for } r = 1 \text{ to } R \text{ do} \\
\quad \text{for } j = 1 \text{ to } N \text{ do} \\
\quad \quad w^{(i+1)} = \min_w \|w - w^i\| \\
\quad \quad \text{s.t. } S(X_t, Y_t) - S(X_t, Y_h) \geq L(Y_t, Y_h) \\
\quad \quad \text{where } Y_h = \text{best}(X_t; w^i) \\
\quad \quad v = v + w^{(i+1)} \\
\quad \quad i = i + 1 \\
\quad w = v/(N \times R)
\]

As one can see from the algorithm outlined, each iteration considers only one datapoint in the training set, and adjusts the weights so the score of the correct compression is better than the score of the best compression as hypothesized by the previous set of weights by a margin that is greater than a pre-defined loss function. In our experiments, we define loss as the Levenshtein distance between the correct sequence in the true compression, and the hypothesized compression, with a uniform penalty of 1 for each deletion, insertion and substitution.

**Experimental Results**

We report our results as precision and recall over whether each frame was classified as *relevant* or *not relevant*. As a simple baseline for comparison, we use a GMM classifier that classifies each frame in the test data as relevant or not relevant, based on the models it learns from the training data. In our setting, suppose we have \( c_1 \) frames that belong to class 1, and \( c_2 \) frames that belong
to class 2. Suppose, we have a model that predicts class 1 for \(pc_1\) frames, and \(c\) of these are correct, then for class 1:

\[
\text{Precision} = \frac{c}{pc_1}; \quad \text{Recall} = \frac{c}{c_1}
\]  

(3.17)

While the interpretations will vary depending on the task that this method is applied to, it appears in this case that false negatives are more harmful than false positives, since if something that is relevant is marked as not relevant, then that segment is removed and the AUDs corresponding to it cannot be used in subsequent processing for classification or retrieval. Thus, high recall for the relevant class is a desirable property. Naturally, precision and recall trade off against each other, but a low precision would imply that either we do a poor job of identifying the relevant segments, or that we are not especially selective in choosing the relevant segments. As a result, we will report precision and recall for the relevant class (Prec-Rel and Recall-Rel, respectively) as well as an overall accuracy. Accuracy considers both relevant and not relevant classes, and the number represents how many frames were correctly labeled over the entire test data. Table 3.4 presents the results of our AUD-based compression system, and compares it with GMM-based classifiers.

The improvement in recall of the relevant class using our approach over the baseline GMMs is statistically significant. The performance of the AUD-based and GMM-based approaches are fairly similar in terms of precision, as well as overall accuracy. There is a common trend we observed from our preliminary experiments using both our AUD-compression approach, as well as the GMM approach— as the number of AUDs or GMM components is increased, the recall of the relevant class worsens. It is not immediately clear what the reason for this is, although it could be that the data is coherent enough that a small set of acoustic descriptors (or Gaussian components for the GMM) suffice to model it fairly well. Alternately, perhaps improving the initial segmentation in our AUD-based approach will lead to even better performance.

As described earlier, the key advantage that the AUD-based approach provides in extracting segments is that contiguous chunks of audio are either retained or dropped. Thus, on synthesizing the segments described as relevant by the AUD-based compression approach, the disfluencies are not as apparent as they are in the GMM-classification based approach, where each frame is separately analyzed for potential retention or deletion. It is worth noting that the dropped chunks do occasionally include relevant segments resulting in abrupt, undesirable changes in context. The effect that these errors will have on perception depend heavily on the application. For instance, in generating a condensed version of the game, dropping relevant segments is likely to be quite aggravating for a user. However, if the identification of relevant segments is to be used as a tool to help the user navigate audio files better, then identifying relevant content within small errors can be acceptable as the user can manually change the boundaries to suit himself.

<table>
<thead>
<tr>
<th>System</th>
<th>Prec-Rel</th>
<th>Recall-Rel</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression-64 AUDs</td>
<td>62.6%</td>
<td>63.8%</td>
<td>63.8%</td>
</tr>
<tr>
<td>32 gaussian GMM</td>
<td>63.4%</td>
<td>55.8%</td>
<td>62.7%</td>
</tr>
<tr>
<td>64 gaussian GMM</td>
<td>58.8%</td>
<td>50.8%</td>
<td>58.6%</td>
</tr>
<tr>
<td>128 gaussian GMM</td>
<td>63.0%</td>
<td>41.4%</td>
<td>59.5%</td>
</tr>
</tbody>
</table>

Table 3.4: Results on MED10 test data
Discussion of Results

The technique described for detection of audio could be applied to perform domain specific extraction of relevant segments, useful for generating highlights from sports videos, or to help users of multimodal search systems navigate through search results. We note that the concept of relevance as used in this work, even in limited domains, can be tricky. For instance, users could have obtained the same set of files by searching for *baseball crowd* or *baseball plays*, but the notion of relevance is almost exactly complementary in the two cases, and there could be other such conflicting queries.

3.3 Discussion

In the various applications we described in this chapter, we discussed some characteristics of the AUDs, especially as they pertained to the the applications. In this chapter, we will discuss some more general observations obtained from the analysis of the discovered AUDs and discuss some of the potential shortcomings and brief thoughts on how they may be addressed in the future.

First, we look at the distribution of AUD frequency as we proceed with the learning. Our initialization of the AUDs used similarity between consecutive frames in the audio to decide whether or not they belonged to the same segment, and then clustered these segments together. Each of these clusters is a distinct AUD, and all segments belonging to a cluster are transcribed with this cluster identity. We first investigated the distribution of these segments and how they change with learning iterations.

Figure 3.7 shows the histogram where the Y-axis is the frequency of occurrence of the AUDs. The AUDs are sorted using frequency of occurrence along the X-axis. As the plot shows, the frequencies of occurrence of the AUDs appear to follow a power law Zipf [1932]. This is not surprising since various naturally occurring phenomenon have been shown to follow such power laws Gabaix [1999], Li [1992], but it does indicate that our initialization does produce something that we would expect.

However, we notice some interesting trends over learning iterations. Figure 3.8 shows the sorted frequency of occurrence of the AUDs at intervals in the learning process. The blue line is the same as the one in Figure 3.7 showing the frequencies of the AUDs immediately after initialization. Even though the sorted frequencies of the AUDs appears to follow a power law at
various stages of training, we find that the frequencies of the AUDs have approximately tripled at the end of the training process. We plot the frequencies after 10 and 20 iterations in the figure. We note that this change is linked to the durations spanned by the AUDs. When they were initialized, each AUD on average captured 0.84 seconds of audio. At the end of the training process, the average length of audio spanned by the AUDs was 0.29 seconds.

Anecdottally, we note two other points about the distribution of the AUDs in the learning process, although their implications are not clear to us at this point. First, the unigram distribution of the AUDs changes significantly over time, as shown in Figure 3.9. Second, the bigram distribution of the AUDs shown in Figure 3.10 also changes significantly over time, although most of the probability mass appears to concentrate along the main diagonal, which implies that states are most likely to transition to themselves. The transition matrix also appears to get more sparse over time, implying that AUDs are most likely to occur in a small number of contexts.
Chapter 4

Proposed Work

In our preliminary work, we described our approaches to modeling the AUD layer in the structure shown in Figure 1.1. In this chapter, we outline the research questions that will be explored in the rest of this thesis, and provide an overview of the approaches we expect to use. Our future work in this thesis will focus on the higher layers— the event and sub-event layers and event dependency modeling— from Figure 1.1. In Section 4.1, we describe our proposed models and approaches.

A major bottleneck in extracting structured information automatically from audio data is the absence of rich annotations in the data that can allow learning in supervised settings. Typically, annotated audio data is weakly supervised, if at all, with labels or keywords for the entire audio file. Due to variability and dynamic nature of the actual audio content, the keywords likely apply to specific segments of the audio, although such detailed annotation is usually not available, and typically, expensive to obtain. This necessitates the modeling of the kind of structure we propose with latent variable models, and we have used a similar approach in modeling the AUD layer. Further, we propose to explore methods in order to leverage the weak supervision available while developing structured models. In Section 4.2, we discuss our proposed model and approaches.

4.1 Structured Event Sequences

One of the advantages of our formulation using AUDs is that it provides us with a mapping from the continuous acoustic space to a discrete, semantically meaningful symbol space. The problem of discovering structure in this discrete symbol space can be considered to be analogous to that of extracting structured latent information from text data. Some of the common problems in text parsing involve representing structure in a similar hierarchical manner to the one we propose. As an example, consider the structure shown in Figure 4.1. In the figure, one can consider the words to be analogous to the AUDs, since both are discrete units and carry semantic information. The audio events are analogous to phrases in text.

Various grammar formalisms for text have been studied extensively. Sound, however, is fundamentally different from text in that the observation sequence is not discrete (as words in text are), and a transcription of the continuous sound into discrete units (AUDs, in our case) is likely to be noisy (as has been shown by work on parsing transcribed speech Charniak and Johnson [2001]). The use of $n$-best lists for sound transcription with AUDs may help obtain the best decodes available under the current models, but such transcription errors are unavoidable with automatic recognition-based systems. As a result, we expect that the process of converting the audio into discrete AUD sequences will result in noisy transcriptions, and these errors will
propagate as we try to use the noisy AUD sequences to discover structural dependencies in audio. While we do still borrow from various techniques used in parsing text, we note that the basic units we work on in case of audio would be analogous to working with noisy text.

Typically, grammars are used to model the deeper structure in the text for various applications such as machine translation, while for other tasks such as topic modeling, information retrieval, shallow features derived over the observed words prove to be sufficient. Event and sub-event information can be considered to be *deep* features for audio. Unlike text, where the observed words are abstracted to parts of speech, and then to phrases, the hierarchical structure in Fig 1.1 abstracts the AUD sequences to events and sub-events that would be observed by humans listening to the audio.

Let us now attempt to define a grammar for an audio dataset. The dataset contains a set of audio classes, $C$. Each class contains a set of audio events, some of which may be shared between classes, resulting in $V$, a set of all the possible events in the dataset. (For simplicity, in this example, we simply consider one set of events, instead of distinguishing between events and sub-events.) Finally, we have a set of AUDs, used to describe the data, $A$.

Thus, our generative model has a prior distribution over the classes, $G_c$, $\forall c \in C$. Each class $c$ is characterized by a distribution over events, $G_{cv}$, $\forall v \in V$, and events have their own distribution over the AUDs, $G_{va}$, $\forall a \in A$. The grammar for the dataset ($D$) is a composition of all 3 layers:

$$G_D = G_c \circ G_{cv} \circ G_{va}$$ (4.1)

We note 2 points about the above formulation– first, the representation of event distributions given class (and AUDs given event) can be conditioned on more than just the class (event), using information about neighboring events (AUDs). Second, we represent the grammar for the dataset as a combination of grammars over classes, events and AUDs, here and elsewhere, for illustration. As mentioned earlier, the AUDs map stochastically to audio frames which are the true observed data, and the sequences of AUDs used are likely to be noisy. The framework also allows us to create more complex hierarchical structures, if necessary, and append the distribution for the added layer of complexity to the grammar $G_D$. The sub-grammars themselves can be useful for a number of applications– consider the distribution of AUDs given events, $G_{va}$. For a retrieval
task, where we are required to recover instances of a particular event, we can use the grammar for the event to detect such instances and return them\(^1\).

Fig 4.2 shows an example of a simple generative process, using the parameters of the grammar at each level. It assumes that units in the same level are independent of each other. In general, however, such a structure would be far too simplistic to model real world data. As an example, Fig 4.3 shows an example of the kind of structure we may expect in baseball data. The tree consists of 4 levels with the root being the class of the data. The next level represents complex events (plays, in the case of baseball) composed of smaller sub-events. It assumes that the complex events are independent in this case (which while not totally true, is a reasonable approximation to simplify the tree structure). It is clear, however, that the various sub-events\((v_i)\) that compose a play contain dependencies. Further, the chunk sequence of AUDs that compose a sub-event are also likely to contain significant dependencies.

While independence assumptions at the audio event level might be suitable for some application domains, such as baseball, they may not hold for domains that are less restricted. Consider an example of the kinds of sound sequences one might hear in movies, as shown in Fig 4.4. In such scenarios, the individual complex events (sniper shot and exchange of gunfire between police and the sniper) are clearly related, since subsequent scenes and the sounds and actions connect to past events. In such cases, while (complex) events are still described by sequences of sub-events as shown by the segmentation in Fig 4.4, it may be simpler to model the relationship between the events by a dependency grammar.

Sound events, especially in movies because of the way the sequence of events unfolds on-screen, will often have long distance dependencies. Efforts to represent such event sequences with a dependency tree will often result in non-projective trees McDonald et al. [2005], Wang and Harper [2004], similar to their counterparts in text, which will need to be handled using approximate algorithms. As an example, consider the orange arc from \(v_5 \) to \(v_1\) (instead of to \(v_4\)). That would make the dependency tree in Fig 4.4 non-projective.

Further, as a result of working over the noisy AUD sequences, there are likely to be segments of the audio where such dependency relationships cannot be discovered with any appreciable

\(^1\) In our prior work Chaudhuri and Raj [2011b], we attempt to do something similar—detecting baseball-related segments in audio, but without an explicitly defined hierarchical structure or notion of events, which could have been used as features.
Figure 4.4: Dependency tree for a sequence in a movie. The orange edge represents an alternative tree where the event $v_5$ is dependent on $v_1$ instead of $v_4$, resulting a non-projective tree.

level of accuracy, resulting in potentially disjoint networks in the dependency graph structure that is discovered over audio events. Nonetheless, we would like to investigate the results of using features derived over the partial graphs for various audio tasks.

Note also that the information from the structure may be used in different ways. Given an audio file, we can use our grammar to build a (most likely) tree. This tree contains information about events, (potentially, sub-events as well) and AUDs, and their sequences. This information could be used to encode a feature set for different applications such as document level classification or detection of different event types. The layer of events in such a tree can be used to provide a segmentation for the data. Information about the dependencies between events could be used to predict future events.

4.2 Approaches to Leveraging Weakly Supervised Data

There are two main issues with learning structured representations of the semantic content in the audio using data. First, richly annotated data, with annotations at various levels providing information of the semantic audio structure, are not readily available. Further, because of the complexity in labeling at multiple granularities, such annotations would be expensive to obtain using human annotators. Specifically, for an audio sequence $x$, we will almost certainly not have access to hierarchical tree annotations $y$ for such sequences. If the correct $y$ were observed, we could obtain a parameterized model $\theta$ that optimizes the likelihood of obtaining $y$ given $x$, using various criterion such as maximum likelihood estimation, maximum classification accuracy [Juang and Katagiri, 1992], or maximum margin [Crammer and Singer, 2001]. The absence of $y$ makes it infeasible to employ the above methods. In such scenarios, typically EM-based algorithms (and their affinity for local maxima) are applied to sum over the various possible $y$, in order to obtain the parameter set $\theta$.

Second, besides this problem of obtaining hierarchically labeled datasets for sound, there exists a real issue with the supervised data that is available, especially for user generated audio
content. To illustrate this, consider a case of a Youtube video of *dog barking*. As one can imagine, such audio often consists of other acoustic events (people talking, traffic sounds, etc) before or after the actual barking. Learning algorithms, however, cannot distinguish between segments that include and don’t include barking within a single file, and assumes that the entire audio represents positive examples of barking. This assumption will reduce the discriminative ability of the model.

The negative class of the *dog barking* data however contains purely negative samples. Thus, we have a set of pure data belonging to one class, but a mixed set in the other class. In the example above, however, we do have the knowledge that the *dog barking* labeled audio must have at least one segment in it that corresponds to a dog barking. Learning algorithms need to be suitably modified to take advantage of this knowledge to train stronger classifiers, and this constitutes one of my current directions of work.

This framework, however, can be generalized to a task of learning from weak supervision, where the system can deduce finer level information from coarse labels– we have bags of data, and a few of those bags are labeled as containing data belonging to class $c_1$, and some as not containing data from $c_1$. We need to learn a model to detect $c_1$, so that we can detect other bags that contain $c_1$, as well as locate the instances. The ability to do this would be useful for a variety of tasks since coarse labels are easier to assign for human annotators, and the system could automatically extract finer level information from them, by locating segments in the larger bag that correspond to the label class. This paradigm is known in the literature as multiple-instance learning Dietterich et al. [1997].

We first suggest some approaches that could be used for modeling events and event dependencies in this setting in Section 4.2.1. We then discuss the problem of using noisy transcriptions in terms of AUDs, and a suggested means of addressing it in Section 4.2.2. Finally, we briefly discuss approaches that can utilize techniques used in multiple instance and active learning frameworks for our task in Section 4.2.3.

### 4.2.1 Approaches to Learning Sound Structures

Before we describe the general framework of our approach, we should note that it would be heavily influenced by the task at hand. An approach for segmentation of an audio stream would proceed very differently to one that attempts to identify or recount event sequences. In general, we adopt an approach based on contrastive estimation Smith and Eisner [2005] that defines a neighborhood-based means to exploit implicit negative evidence, using likelihood based objective functions that generalize the one maximized by the EM algorithm. Below, we explain our formulation in the context of an event detection problem, specifically focusing on obtaining maximum leverage from the precious little supervision we have.

Suppose we have a small amount of labeled data, from audio sequences marked as positive examples of the event we are attempting to detect. As outlined in Smith and Eisner [2005], part of the reason EM-based approaches might work poorly on such tasks is that while the positive examples tell the algorithm to assign higher probabilities to the positive observed cases, there is no notion of where the probability mass comes from, in the absence of negative examples (implying that the entire remaining space is treated to be negative). The contrastive estimation approach seeks to define a neighborhood of implicit negative evidence around the positive observation, since a user chose the segment as a positive instance from its neighborhood, and outline algorithms for training the parameters in the supervised and unsupervised cases. Thus, in our task of detecting events from AUD sequences, we can use the labeled observation span as a positive sequence.
instance and other sequences in the neighborhood as negatives.

As is apparent from this example, different tasks will require different neighborhood functions—and perhaps, each task needs to experiment with different neighborhood functions. For event detection, we suggested the use of neighboring AUDs within a window as a possible neighborhood. Instead, one may wish to model the problem differently and treat events as words with AUD sequences being letters or phones, and treating other AUD sequences within a Levenshtein distance window as being the neighborhood.

Instead of working in the probability space, as in the case of contrastive estimation based approach, we could also operate with scores in a graph search framework, not constrained by probabilities. The problem of obtaining a dependency tree from discrete symbol sequences has been shown to be equivalent to finding maximum spanning trees in directed graphs McDonald et al. [2005]. In order to do this, we use a directed graph where we represent the symbol sequence as a set of nodes and an edge from node \(i\) to node \(j\) indicates that \(i\) depends on \(j\). The score of the final dependency tree is factorized to be the sum of the edge scores in the final tree. The score of an edge can represented as a weighted combination of features derived over the edge, including information about the adjacent nodes for the edge, context information for those nodes, and temporal information about their occurrence.

The score of a dependency tree \(y\) for a sequence \(x\) is given by:

\[
s(x, y) = \sum_{(i,j) \in y} s(i, j) \tag{4.2}
\]

Scores on edges between nodes are given by a weighted combination of feature values for the edge (\(\theta\) is a weighting scheme over the features, while \(f\) represents the feature vector over an edge):

\[
s(i, j) = \theta \cdot f(i, j) \tag{4.3}
\]

Once scores have been computed for all possible edges in the graph, we can employ the Chu-Liu-Edmonds Chu and Liu [1965], Edmonds [1967] algorithm to find the maximum spanning tree, which searches the entire space of all possible tree with no restrictions, and can work in the non-projective scenario.

The final hurdle lies in obtaining the weight vector \(\theta\). For the task of obtaining dependency trees for sentences in text, McDonald et al. McDonald et al. [2005] describe a large margin learning algorithm Crammer and Singer [2003] in the training phase to obtain the weight vector. This algorithm attempts to learn the weight vector such that for any training instance, the score of the correct dependency tree is better than any incorrect dependency tree by a margin proportional to the degree of incorrectness:

\[
s(x, y_t) - s(x, y') \geq k \cdot L(y_t, y'), \forall y' \tag{4.4}
\]

In the above equation, the function \(L\) is the loss function that measures the difference between the ground truth \(y_t\) and the hypothesized tree \(y'\).
4.2.2 The Problem of Noisy Transcriptions

Thus far we’ve referred to various representations of the structure that we expect to infer from the audio. Although we have mentioned that the mapping between the acoustics of a recording and the underlying sequence of AUDs is stochastic, in the discussion above we have largely treated the AUDs as known. The reality of the situation is otherwise, and the true relation between the various variables in the model are as shown in the abstract model of Fig 4.5. In the figure the square, shaded blocks represent the observed audio data ($X$), the connected circles in the center represent the underlying sequence of AUDs, and the circle labelled $G$ represents the governing grammar over the AUDs. The AUDs themselves are latent – they are not observed and can only be inferred.

Note that in the model, as shown in Fig 4.5 the AUD sequence isolates the audio $X$ from the grammar $G$; $G$ governs $X$ only through the AUD sequences it can generate. The model can therefore be factored into two layers: $L_1$, which represents the dependence between the grammar $G$ and the AUD sequence $a$, and $L_2$, which shows the dependence between $a$ and $X$. The text so far has discussed our ideas primarily through the $L_1$ layer. We now discuss mechanisms by which the coupling between $L_1$ and $L_2$ may be handled.

Let $\alpha$ represent the set of parameters through which $G$ generates $a$. Let $\beta$ represent the parameters through which $a$ generates $X$. In order to learn all the parameters of the model both $\alpha$ and $\beta$ must be learned. In order to make inferences over novel data using the model both $\alpha$ and $\beta$ must be known.

In order to train the system shown in Fig 4.5, we would need to jointly estimate the parameters $\alpha$ and $\beta$, by optimizing an objective function as below:

\[ \text{Note that the score is not the probability of the tree} \]
arg max \( P(X|\alpha, \beta) = \arg \max_{\alpha, \beta} \sum_a P(X, a|\alpha, \beta) \) (4.5)

Here, since the AUDs sequence \( a \) is an unobserved latent variable, we must integrate over all possible values of \( a \).

Directly learning (or performing inference over) the above model can be arbitrarily difficult, depending on the complexity of the assumed underlying grammar \( G \). In order to simplify the estimation variational approximations may be employed to decouple the two layers as shown in Fig 4.6. However, both learning and inference can remain complex.

An alternate formulation modifies the objective function to explicitly also estimate the optimal AUD sequence. In this case the estimation gets modified to

\[
\arg \max_{\alpha, \beta, a} P(X, a|\alpha, \beta) \quad (4.6)
\]

This now enables decoupling of the two layers of the model to the graphical structure shown in Fig 4.6 as follows:

\[
\arg \max_a P(X, a|\alpha, \beta) \quad (4.7)
\]

\[
\arg \max_\beta P(X, a|\beta) \quad (4.8)
\]

\[
\arg \max_\alpha P(a|\alpha) \quad (4.9)
\]

Note that Equation 4.7 represents the estimation of the optimal AUD sequence given all parameters of the model in Fig 4.5, which is generally a tractable problem. Equation 4.8 represents estimation of the parameters of the \( L_2 \) layer in the decoupled model of Fig 4.6, and Equation 4.9 represents the estimation of the parameters of the \( L_1 \) layer.

This set of iterations leads to a simpler formulation, although the EM-like estimation process could potentially lead to local optimum solutions. This formulation also depends on the best decode of the audio based on the parameters, thus introducing noise which further propagates since the parameters are updated based on the decode.

Yet another formulation, which may be expected to be less likely to arrive at poor local optima employs importance sampling over \( a \), where the samples may be obtained through \( n \)-best decodes of the audio into AUD sequences. Here we no longer explicitly estimate the best AUD sequence; computational tractability is achieved by considering only a limited but representative set of AUD sequences, rather than all possible AUD sequences. The resulting estimates may be obtained as weighted combinations of the estimates obtained from the individual AUD sequences. Let \( I(a_i) \) be the importance weight of the \( i \)-th AUD sequence from the \( n \)-best list of AUD decodes, given the observed audio and the previous estimates of the parameters.

If \( \alpha_i \) and \( \beta_i \) are the corresponding parameters obtained, using the \( i \)-th decode by following the estimation procedure outlined in Equation 4.6, then we can average this parameters to generate the parameter set:

\[
\alpha^* = \frac{\sum_{i=1}^n I(a_i)\alpha_i}{\sum_{i=1}^n I(a_i)} \quad (4.10)
\]

\[
\beta^* = \frac{\sum_{i=1}^n I(a_i)\beta_i}{\sum_{i=1}^n I(a_i)} \quad (4.11)
\]
The effectiveness of the various formulations will depend on the grammar structure employed to govern the AUD sequences, and potentially, on the task it is being applied to as well. In general, if the grammar were to impose a higher degree of structure, we can employ simpler formulations, since the structure would guide the solution to a greater degree.

4.2.3 Weakly Supervised Learning Approaches

The rich hierarchical structure we propose to infer from audio does not exist in any standard audio datasets. Further, while various audio data sets do contain informative labels, they often apply only at specific (and usually unknown) granularities. The ability to infer labels from unlabeled data using small amounts of labeled data, as well as understanding the granularities at which such labels apply, automatically would make audio analysis systems more powerful.

Consider an example from the MajorMiner dataset Maj [2007]. The dataset contains labels for clips of music where the labels correspond to various kinds of information relevant to the music, including genre, mood, tempo, etc. Musical pieces will often capture multiple moods in the same piece. As a result, the annotations might contain multiple mood labels for the same piece. Typically, supervised approaches treat the entire audio file as a positive instance for each mood label associated with the file, resulting in a weakened ability of the learnt models to discriminate between positive and negative instances. This issue sheds some light on the level of difficulty in the annotation process—asking human users to navigate through clips to label individual segments to generate richer annotations results in a harder annotation task, which is more expensive in terms of time and cognitive effort. On the other hand, asking them to listen to an entire audio clip and annotating it with keywords they think apply to the clip is significantly easier.

One way of handling this issue would be to attempt to automatically assign the labels to their correct granularity, where they are directly applicable. In such a setting, we can treat the audio file as a bag, and divide the audio into segments of audio. These segments are treated as individual data points contained in the bag. Let us consider the case of building a detector for a specific label: various audio files have been annotated with that label and are positive instances, while the remaining files in the dataset are negative instances. Each instance can be represented as a bag of data points, where each datapoint is a segment of the audio.
Thus, the training setup will appear to be as shown in Fig 4.7. The quadrilaterals represent bags of data—red balls represent the audio segments that are negative instances of the label, and blue balls represent positive instance. Thus, the positive bags are guaranteed to contain at least one positive data point, while the negative bags contain no positive data points. Such a learning setting is known as multi-instance learning in the literature Dietterich et al. [1997]. A standard approach to solving this problem utilizes a mixed-integer linear program in a Support Vector Machine framework, where the problem can be represented as follows:

$$\min_{y_{ij}, w, b, \xi} \frac{1}{2}||w||_2^2 + C \sum_{ij} \xi_{ij}$$ \hspace{1cm} (4.12)

subject to

$$y_{ij}(w.x_{ij} + b) \geq 1 - \xi_{ij}$$ \hspace{1cm} (4.13)

$$y_{ij} \in \{-1, 1\}$$ \hspace{1cm} (4.14)

$$\xi_{ij} \geq 0$$ \hspace{1cm} (4.15)

$$\sum_{j=1}^{l_i} \frac{1}{2}(y_{ij} + 1) \geq 1, \forall i \in I^+$$ \hspace{1cm} (4.16)

$$y_{ij} = -1, \forall i \in I^-$$ \hspace{1cm} (4.17)

In the above, $i$ indexes the bags of data, while $j$ indexes the data points in the bag. $I^+$ indicates the positive bags while $I^-$ indicate negative bags. The number of instances in the $i$-th bag is assumed to be $l_i$.

The approach described above is one instance of a solution for this learning setting. Various other frameworks, including boosting, MCMC sampling, etc, have been developed for multi-instance learning. The advantages of using this approach is two-fold: first, it would enable us to leverage coarse annotations to automatically infer their correct segments of applicability. Second, the ability to accurately label segments of audio pieces provides a natural mechanism for providing enhanced navigability to users.

This, however, leads us to a further and more difficult learning setting. Consider a scenario where only a small number of annotators are available, and the entire large dataset cannot be scanned by them within a reasonable time frame to generate only the coarse labels. In this setting, we have a small amount of weakly supervised data (the audio files for which coarse labels have been provided) and a large amount of unsupervised data.

In the scenario described above, we can apply many of the techniques described briefly earlier in Section 2.3. First, such a learning setting is one where semi-supervised learning techniques apply directly. Instead of relying on a wholly automated process, however, it likely makes sense to obtain expert annotations from humans to obtain labels. The process of selecting data points to be annotated can be guided by active learning approaches, where the objective is to select instances in a manner that will allow the learner to benefit as much as possible in learning a more accurate model. It has been noted in the literature that the process of obtaining labels for specific examples is fraught with various issues in the real world— the annotator may be wrong, unavailable, uncertain or too expensive. For the specific task of providing rich, structured labels, we earlier suggested that the expense in time and effort may be significant, and could affect accuracy of such labels. In order to reduce such expense, one might explore different active learning settings— e.g. where the user simply confirms whether a predicted label (that the model
is uncertain about) is correct or not, instead of having to generate the label. Methods to relax the assumptions made by active learning have been explored in the literature and are referred to as proactive learning Donmez and Carbonell [2008], and could be explored for our tasks as well.

### 4.3 Future Tasks and Estimated Timeline

In Section 4.1 and Section 4.2, we outlined approaches that could be used to infer event structure over AUDs. This involves 2 related tasks:

- **Modeling events as sequences of AUDs:** This portion of the work attempts to model the events layer shown in Figure 1.1. We will model events as sequences of AUDs potentially leveraging information from annotations, provided with the data.

- **Dependency and relationships between events:** Here, we intend to analyze our inferred event sequences, using information from both the event sequence itself as well as the AUDs layer, to predict dependencies between events, where possible. A dependency between 2 events says that our likelihood of seeing one increases when we see the other, and may capture temporal relationships of their occurrence patterns as well.

- **Investigation of Semi-Supervised Approaches for data-driven event discovery:** Various semi-supervised approaches can be used to leverage the labels that we have available. Multi-instance learning and active learning are 2 specific examples of approaches that could be used to enable the system to better use the limited annotations available.

- **Applying hierarchical features to audio processing tasks:** Our hypothesis is that being able to infer event structure from audio automatically will help us perform better on various audio processing tasks that should benefit from the use of semantic knowledge. We will evaluate feature sets that incorporate features from the hierarchical analysis on tasks such as audio classification, audio retrieval, audio event detection.

- **Optional – Structure Discovery in Constrained Domain:** We are interested in using this hierarchical approach to model real-world data. We evaluate performance on various audio processing tasks, since the lack of clearly annotated data doesn’t allow us to evaluate performance on the correctness of the inferred structures. We propose to evaluate the structures on data from a domain where clear structure is expected to be present, such as music.

We plan to make progress towards the proposed work using the timeline in Table 4.1 as a guide.

<table>
<thead>
<tr>
<th>Expected Time</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov, 2011 – Jan, 2012</td>
<td>Modeling of the event layer</td>
</tr>
<tr>
<td>Jan, 2012 – Apr, 2012</td>
<td>Modeling dependencies and relationships between events</td>
</tr>
<tr>
<td>May, 2012 – Oct, 2012</td>
<td>Semi-supervised approaches to leveraging weak supervision</td>
</tr>
<tr>
<td>Nov, 2012 – Jan, 2013</td>
<td>Evaluating event-augmented feature sets for various applications</td>
</tr>
<tr>
<td>Apr, 2013</td>
<td>Dissertation defense</td>
</tr>
</tbody>
</table>

Table 4.1: Estimated timeline for progress toward thesis completion
Bibliography

http://www.sound-ideas.com/artfoley.html. 2005. 3.2.3
http://majorminer.org/info/intro. 2007. 4.2.3
http://www.nist.gov/itl/iad/mig/med11.cfm, 2011. 3.2.1
M. Bacchiani. Speech Recognition System Design Based on Automatically Derived Units. *PhD Thesis*, 1999. 3.1
S. Chaudhuri, R. Singh, and B. Raj. Data Driven Acoustic Units for Audio Classification. In submitted to ICASSP, 2012a. 3.2.3
P.P. Cruz-Alcazar and E. Vidal. Two Grammatical Inference Applications in Music Processing. Applied Artificial Intelligence, 2008. 2.1
P.F. Felzenszwalb and D. McAllester. Computer Science Technical Report-2010-02,, 2010. 2.4
Retrieval Track Overview and Results. In Proceedings of TREC-6 Conference, 1997. 2.1
S. Kim, P.G. Georgiou, and S. Narayanan. Supervised acoustic topic model for unstructured audio information retrieval. In Proceedings of Asia Pacific Signal and Information Processing Association (APSIPA) Annual Summit and Conference, 2010. 3.2.4
K. Lee and D. Ellis. Audio-Based Semantic Concept Classification for Consumer Video. IEEE Transactions on Audio, Speech and Language Processing, 18:1406–1416, 2010. 2.1
Z. Liu, J. Huang, Y. Wang, and T. Chen. Audio Feature Extraction and Analysis for Scene Classification. In Workshop on Multimedia Signal Processing, 1997. 2.1

43
R. McDonald. Discriminative Sentence Compression with Soft Syntactic Evidence. In *Proceedings of European Association for Computational Linguistics*, 2006. 3.2.4


R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Association for Computational Linguistics*, 2005. 4.1, 4.2.1, 4.2.1

M. McKinney and J. Breebaart. Features for Audio and Music Classification. In *International Symposium on Music Information Retrieval*, 2003. 3.2.3


B. Settles. Active Learning Literature survey, 1995. 2.3


M. Slaney. Mixture of probability experts for audio retrieval and indexing. In *ICME*, 2002a. 2.1


N.A. Smith and J. Eisner. Guiding Unsupervised Grammar Induction Using Contrastive E-


E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based Classification, Search and Retrieval of Audio. In *IEEE Multimedia Magazine*, 1996. 1, 2.1


