UNIVERSITY OF CALIFORNIA, SAN DIEGO

Machines that Understand Music

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Electrical Engineering (Signal and Image Processing)

by

Luke Barrington

Committee in charge:

Professor Gert Lanckriet, Chair
Professor Garrison W. Cottrell
Professor Bhaskar Rao
Professor Lawrence Saul
Professor Nuno Vasconcelos

2012

The dissertation of Luke Barrington is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2012

DEDICATION

To my mum and dad, for getting me here...

and to Laia, for keeping me going.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I thank the following people who, in one way or another, made this work possible:

Damien O'Malley, for design, development and testing of Herd It, for invaluable "board" meetings and for helping me avoid turning completely septic.

Brian McFee for teaching me how to hack, for fueling me with the funk and for his boundless, unrelenting and always uplifting cynicism.

Antoni Chan for ideas, code, tunes, conversations, parties, lessons, recipes, trips and basslines. Thanks a ton.

Douglas Turnbull for blazing the trail that I could stroll through.

Albert Lin, Nate Ricklin and Shay Har-Noy for continuing crazy adventures and for the support and ambition to get the hell out of here. *Fiiiiiiiiiiiiles*.

Michael Lyons for showing me Kyoto and teaching me that messing with music really could be the subject of an engineering Ph.D thesis.

Garrison W.Cottrell for getting me started and then letting me go but supporting me all the way. Thanks, dude.

Robert Hecht-Nielsen for giving me the inspiration to pursue greatness and the conviction to believe that I deserve it.

Gert Lanckriet for winding me up and (finally) setting me free.

Soren Solari and Cecile Levasseur for getting me past those first two years.

Patrick Amihood, Chandra Murthy and David Wipf, may we surf together forever.

All gurons, especially Matt Tong.

All of my Computer Audition Laboratory, especially Emanuele "Mario" Coviello who carries on the torch, higher and brighter.

Arshia Cont, Sanjoy Dasgupta, Shlomo Dubnov, Charles Elkan, Ken Kreutz-Delgado, Andrew Huynh, Reid Oda, Miller Puckette, Bhaskar Rao, Lawrence Saul, David Torres, David Vanoni, Nuno Vasconcelos and all the professors, students, researchers and staff at UCSD who tolerated, taught and inspired me through eight great years.

Chapter 2, in full, has been submitted for publication of the material as it may appear in Proceedings of the National Academy of Science, 2011, L. Barrington, D. Turnbull and G.R.G. Lanckriet. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in IEEE Transactions on Acoustics, Speech and Language Processing, 16(2) pp467-476, 2008, D. Turnbull, L. Barrington, D. Torres and G.R.G. Lanckriet. The dissertation author was investigator and co-author of this paper.

Chapter 4, in full, is a reprint of the material as it appears in IEEE Transactions on Acoustics, Speech and Language Processing, 18(3) pp602-612, 2010, L. Barrington, A.B. Chan and G.R.G. Lanckriet. The dissertation author was the primary investigator and author of this paper.

VITA

2001	Bachelor of Electronic Engineering University College Dublin, Ireland
2004	Master of Science Electrical Engineering (Signal and Image Processing) University of California, San Diego
2012	Doctor of Philosophy Electrical Engineering (Signal and Image Processing) University of California, San Diego

PUBLICATIONS

L. Barrington, D. Turnbull and G.R.G. Lanckriet. Game-Powered Machine Learning. *Proceedings of the National Academy of Sciences*, in press, 2012.

B. McFee, L. Barrington, and G.R.G. Lanckriet. Learning Similarity from Collaborative Filters. In *International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010.

E. Coviello, L. Barrington, G.R.G. Lanckriet and A.B. Chan. Automatic Music Tagging with Time Series Models. In *International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010.

L. Barrington, A.B. Chan and G.R.G. Lanckriet. Modeling Music as a Dynamic Texture. *IEEE Transactions on Audio, Speech and Language Processing*, 18(3) 602-612, 2010.

L. Barrington, R. Oda and G.R.G. Lanckriet. Smarter Than Genius? Human Evaluation of Music Recommender Systems. In *International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

L. Barrington, D. Turnbull, D. O'Malley and G.R.G. Lanckriet. User-Centered Design of a Social Game to Tag Music. In *Workshop on Human Computation*, Paris, France, 2009.

L. Barrington, D. Turnbull, M. Yazdani and G.R.G. Lanckriet. Combining Audio Content and Social Context for Semantic Music Discovery. In *Special Interest Group on Information Retrieval*, Boston, MA, 2009

L. Barrington, A.B. Chan and G.R.G. Lanckriet. Dynamic Texture Models of Music. In *International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, 2009.

L. Barrington, T.K. Marks, J.H.W. Hsiao and Cottrell. NIMBLE: A Kernel Density Model of Saccade-Based Visual Memory. *Journal of Vision*, 8(14):17, 1-14, 2008.

L. Barrington, M. Yazdani, D. Turnbull and G.R.G. Lanckriet. Combination of Feature Kernels for Semantic Music Retrieval. In *International Society for Music Information Retrieval Conference*, Philadelphia, PA, 2008.

D. Turnbull, L. Barrington and G.R.G. Lanckriet. Five Approaches to Collecting Tags for Music. In *International Society for Music Information Retrieval Conference*, Philadelphia, PA, 2008.

D. Turnbull, L. Barrington, D. Torres and G.R.G. Lanckriet. Semantic Annotation and Retrieval of Music and Sound Effects. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2), 467-476, 2008.

D. Torres, D. Turnbull, L. Barrington and G.R.G. Lanckriet 2007. Identifying Words that are Musically Meaningful. In *International Society for Music Information Retrieval Conference*, Vienna, Austria, 2007.

D. Turnbull, R. Liu, L. Barrington and G.R.G. Lanckriet. A Game-Based Approach for Collecting Semantic Annotations of Music. In *International Society for Music Information Retrieval Conference*, Vienna, Austria, 2007.

D. Turnbull, L. Barrington, D. Torres and G.R.G. Lanckriet. Towards Musical Query-by-Semantic-Description using the CAL500 Data Set. In *ACM Special Interest Group on Information Retrieval*, Amsterdam, Netherlands, 2007.

L. Barrington, T.K. Marks and G.W. Cottrell. NIMBLE: A Kernel Density Model of Saccade-Based Visual Memory. In *Cognitive Science Society Conference*, Nashville, TN, 2007.

L. Barrington, A.B. Chan, D. Turnbull and G.R.G. Lanckriet. Audio Information Retrieval Using Semantic Similarity. In *International Conference on Acoustics, Speech and Signal Processing*, Honolulu, HI, 2007.

D. Turnbull, L. Barrington and G.R.G. Lanckriet Modeling Music and Words. In *International Society for Music Information Retrieval Conference*, Victoria, Canada, 2006.

L. Barrington and G.W. Cottrell. Automatic Visual Integration - Defragmenting the Face. In *Cognitive Science Society Conference*, Vancouver, Canada, 2006.

L. Barrington, M. Lyons, D. Diegmann and S. Abe. Ambient Display Using Musical Effects. In *Intelligent User Interfaces*, Sydney, Australia, 2006.

ABSTRACT OF THE DISSERTATION

Machines that Understand Music

by

Luke Barrington
Doctor of Philosophy in Electrical Engineering
(Signal and Image Processing)
University of California, San Diego, 2012
Professor Gert Lanckriet, Chair

Machine learning, signal processing and data mining are being combined to analyze audio content in a relatively new field of research called *computer audition*. This thesis develops and describes a number of computer audition methods and shows how they can be applied to solve challenges including automatic tagging, similarity and recommendation, search and discovery, and segmentation of music content. To achieve these advances in music understanding requires human guidance. A further contribution of this work is to pioneer *game-powered machine learning* that uses crowdsourced human intelligence to guide the training of machine algorithms. By leveraging human perception with machine automation, the work described in this thesis presents a comprehensive approach to computer audition that leads to the development of machines that understand music.

Chapter I

Introduction

The explosive rise of music creation, distribution and consumption in digital formats has seen the emergence of a new field of research called *computer audition*. Sometimes termed "music information retrieval" (MIR) or "machine listening", computer audition aims to advance automated methods for analyzing and, ultimately, understanding music *content*: that is to "listen" to an music waveform and extract useful information. The specific type of information, and the uses to which it is put, range across a wide variety of application areas including song similarity, music recommendation, tempo estimation, song structure analysis and semantic music tagging i.e., description of music with relevant textual metadata, including artist names, genres, emotions, instruments and more.

This thesis describes a number of computer audition methods and investigates their role in the aforementioned applications. The computer audition advances reported here – as well as the long list of related work from colleagues in the fields of MIR, machine learning, signal processing, acoustics, speech recognition, computer vision, video processing, time-series analysis, data mining, collaborative filtering, statistics, neuroscience, psychology and more – move towards the development of machines that can indeed understand music. However there is, as yet, no computer substitute that can begin to match the most sophisticated music processing machinery in existence: the human mind. The ability to produce, perceive, identify, analyze, describe, emote with and get down to music is a uniquely human trait (well, almost [86]). Thus, machine understanding can best be derived from human interpretation of music.

For this reason, this thesis proposes a novel method for advancing computer audition that explicitly brings humans into the loop. Termed "Game-Powered Machine Learning" and described in Chapter 2, this method exploits human social networks online to "feed the machine". Using a game called "Herd It" to collect *crowdsourced* human consensus about semantic music interpretations, this game-powered approach provides reliable information required to train machine learning models of music. Chapter 3 continues by describing one of the first machine learning

algorithms for automatically tagging music content, that is describing music with relevant words or "tags". Termed a music *autotagger*, such algorithms can be used to index audio content with semantic terms, powering a applications including music search engines, song similarity and music recommendation and discovery. While the model in Chapter 3 automates many human-like descriptions of music, it's analysis makes a simplifying assumption that ignores the *temporal* aspects of how a piece of music changes from start to end. Chapter 4 takes steps to redress this shortcoming by introducing a *dynamic* model of musical content. By considering both the timbral and temporal aspects of a piece of music, this model can automatically determine the structure of a song and points the way to more descriptive representations of audio content.

With musicians all over the planet creating, recording, remixing and sharing music like never before, we all have access to hundreds of millions of songs with the push of a button. Computer audition and related work offer to help us find the top hits, classic cuts and rare gems in this sea of content. By leveraging advances in machine learning and crowdsourcing the unparalleled expertise of human perception, this thesis describes how computers can describe, segment and recommend audio content and brings us closer to the day when machines can understand music.

Chapter II

Game-Powered Machine Learning

II.A Abstract

Searching for relevant content in a massive amount of multimedia information requires that each image, video, or song be accurately annotated with a large number of relevant semantic keywords, or *tags*. We introduce game-powered machine learning, an integrated approach to annotating multimedia content that combines the effectiveness of *human computation*, through online games, with the scalability of *machine learning*. We investigate this framework for labeling music. First, a socially-oriented music annotation game called *Herd It* collects reliable music annotations based on the "wisdom of the crowds". Second, these annotated examples are used to train a supervised machine learning system. Third, the machine learning system *actively* directs the annotation games to collect new data that will most benefit future model iterations. Once trained, the system can automatically annotate a corpus of music much larger than what could be labeled using human computation alone. Automatically annotated songs can be retrieved based on their semantic relevance to text-based queries (e.g., "funky jazz with saxophone", "spooky electronica", etc.). Based on the results presented in this chapter, we find that actively coupling annotation games with machine learning provides a reliable and scalable approach to making searchable massive amounts of multimedia data.

II.B Introduction

The last decade has seen an explosion in the amount of multimedia content available online: over 3 billion images are uploaded to Facebook each month[1], YouTube users upload 24 hours of video content per minute[2], and iTunes, the world's largest music retailer, offers a growing catalog of more than 13 million songs[3]. Developing a *semantic multimedia search engine* – that enables simple

[1] http://www.facebook.com/press/info.php?statistics
[2] http://www.youtube.com/t/fact_sheet
[3] http://www.apple.com/itunes/features/

discovery of relevant multimedia content as easily as Internet search engines (e.g., Google [16]) help us find relevant web pages – presents a challenge since the domain of the query (text) differs from the range of the search results (images, video, music).

To enable semantic search of non-textual content requires a mapping between multimedia data and a wide vocabulary of descriptive *tags*. Describing multimedia content with relevant semantics necessitates intervention from humans who can understand and interpret the images, video or music. However, manual tagging by human experts is too costly and time-consuming to be applied to billions of data items. For example, Pandora, a popular Internet radio service, employs musicologists to annotate songs with a fixed vocabulary of about five hundred tags. They then create personalized music playlists by finding songs that share a large number of tags with a user-specified seed song. After ten years of effort by up to 50 full-time musicologists, less than 1 million songs have been manually annotated[4], representing less than 8% of the current iTunes catalog.

Crowdsourcing has emerged as an affordable and scalable alternative to expert annotation by engaging many non-expert contributors to label content online. Participants are motivated through small monetary rewards[5], or, even better, to contribute for free by disguising tasks as fun games, appealing to scientific altruism, or requiring it to access a service of interest. This distributed *human computation* has been applied, e.g., to categorize galaxies[6], fold proteins [26], transcribe old books [121], classify smiles [123] and apply descriptive tags to images [120], web pages [119] and music [63] (see Section II.B.1 for a review). Despite the promise of recruiting vast amounts of free labor, human computation *games* have had limited success in tagging the vast amount of multimedia content on the web: in 5 years, the ESPgame [120] has collected labels for up to 100m images — roughly the same number that are uploaded to Facebook daily — and TagATune [63] has labeled

[4] `http://blog.pandora.com/faq/`
[5] e.g., Amazon's Mechanical Turk: `http://mturk.amazon.com`
[6] `http://www.galaxyzoo.org`

30,000 song clips, or about 0.23% of iTunes' catalog.

Instead of requiring that humans manually label every image, video or song, tagging can be partially automated using *supervised machine learning* algorithms that learn how semantics relate to multimedia. Machine learning approaches discover consistent patterns among a modest number of pre-labeled training examples and then generalize this learned knowledge to label new, unlabeled data. The scalability of computer automation offers the potential to categorize massive amounts of multimedia information but reliability hinges on the *quality of training data* used. For example, by learning from millions of example images of faces in all possible poses, angles and lighting conditions, machine learning algorithms [117] rapidly and reliably detect faces to automate focus in consumer digital cameras.

In this chapter, we propose and investigate *game-powered machine learning* as a reliable and scalable long-term solution to annotating large amounts of multimedia content for semantic search, by leveraging the effectiveness of human computation through online games with the scalability of supervised machine learning. The main idea, illustrated for music search in Figure Figure II.1, is to use an online *annotation game* to collect reliable, human-labeled examples that are tailored for training a supervised machine learning system. Once trained, this system can automatically annotate new content with the same tags that are used in the game, rapidly propagating semantic knowledge to lots of multimedia content. Through an *active learning* feedback loop, the games focus on collecting the data that will most effectively improve future machine learning updates.

To validate the effectiveness of game-powered machine learning for *music* search, we designed and developed "Herd It", an online music annotation game that motivates players to contribute tags for songs. In contrast to previous "games with a purpose" which have aimed to annotate every image [120] or song [63, 73] on the web, Herd It was designed with a different, unique and more realistic goal in mind: to enable the active machine learning approach presented in Figure Figure II.1. To this end, Herd It was designed to allow integration with a machine learning

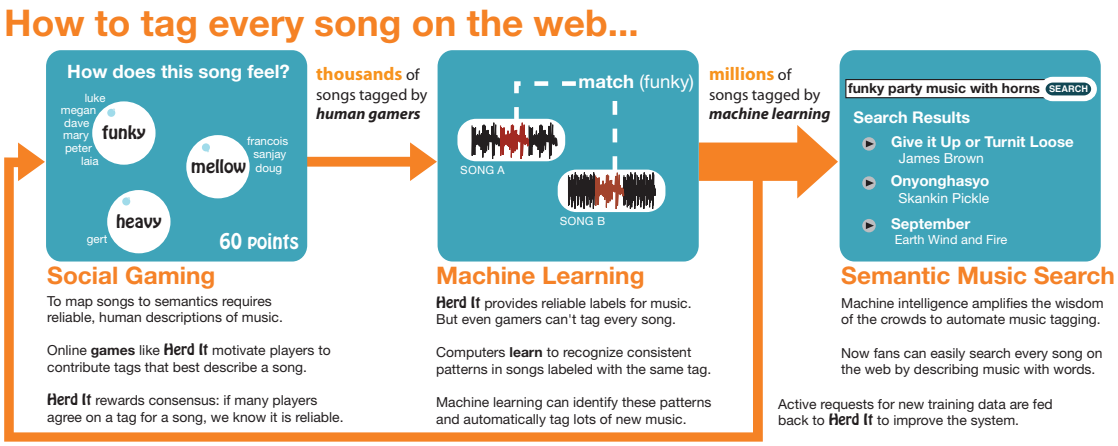


Figure II.1 Game-powered machine learning framework for music annotation.

system that actively suggests training songs and tags, to be presented to users. As a result, the game can collect the most effective data for training the machine learning algorithm that then automates large-scale music tagging. Besides focusing human effort on efficient data collection, active song and tag suggestion also made gameplay more appealing.

We deploy this game-based machine learning system to investigate and answer two important questions. *First*, we demonstrate that the collective wisdom of Herd It's crowd of non-experts can *train* machine learning algorithms as well as expert annotations by paid musicologists. In addition, our approach offers some distinct advantages over training based on static expert annotations: it is cost-effective, scalable, and has the flexibility to model demographic and temporal changes in the semantics of music. *Second*, we show that integrating Herd it in an active learning loop allows to learn accurate tag models more effectively, i.e., with less human effort, compared to a passive approach. We note that the resulting data is publicly available on the Computer Audition Lab's website: `http://cosmal.ucsd.edu/cal/`.

II.B.1 Related Work on Human Computation

Herd It and similar "games with a purpose" [26, 59, 63, 73, 75, 120] offer fun and competition as incentives to motivate wide-spread human participation in scientific endeavors. Other human computation approaches engage participation on a volunteer basis [15, 90, 118] or use Amazon's Mechanical Turk[7] to offer small monetary rewards in return for completing data labeling tasks [4, 78, 79, 95, 103, 104, 116]. The majority of applications have focused on classifying text [4, 15, 103] or images [79, 90, 95, 104, 116, 122] although speech transcription [75, 78] and video labeling [118] applications also exist. Vijayanarasimhan and Grauman [116] estimate the human effort required on Mechanical Turk to obtain image labels of varying granularity (i.e., weak labels for the entire image to explicit object segmentation and tagging) and balance this with the value of information to the resulting discriminative classifier.

Beyond labeling of multimedia data, human computation methods have been applied to numerous fields where the so-called "wisdom of the crowds" provides insight beyond what individual experts can offer. Crowdsourcing successes include prediction markets for sports betting [29], product development scheduling [98], company stocks [105] and political races [13].

Data collected by annotation games has been used to *evaluate* the output of machine learning systems. E.g., data from the ESPgame [120] has been used as a computer vision test set [71] and both MajorMiner [73] and TagATune [63] have been used to evaluate and compare different music tagging algorithms [37, 64]. To date, attempts to use human computation to *train* machine learning systems as accurately as training them from expert data have focused on using Amazon's Mechanical Turk, rather than games. Novotney et al. [78] use Mechanical Turk to crowdsource transcriptions of phone conversations and find a small reduction in performance of the resulting speech recognition system, compared to the same system trained on expert transcriptions. Ambati et al. [4] also use Mechanical

[7] `http://mturk.amazon.com/`

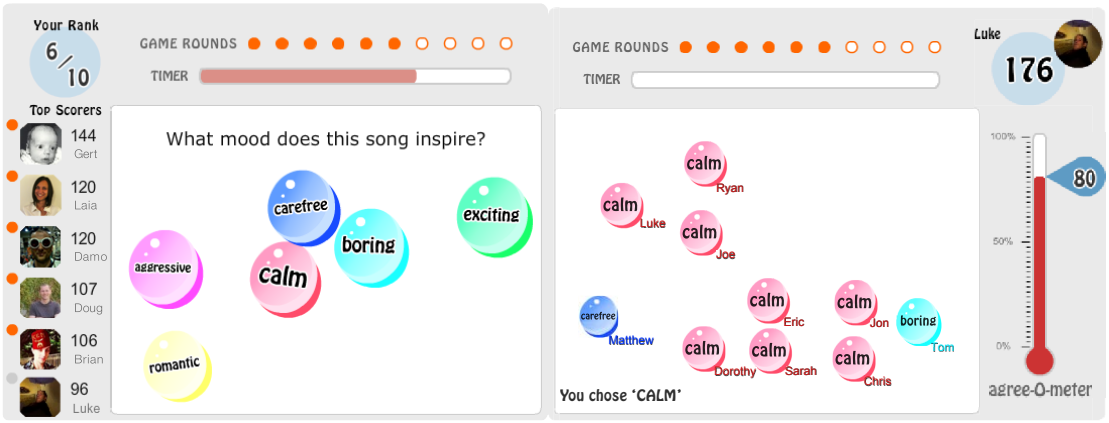
Turk to collect 3,000 English translations of Spanish sentences and train a machine translation system that rivals a system trained on expert annotations.

The proposed game-powered machine learning moves beyond monetary incentives and collects training data for free, a potentially more sustainable and scalable approach. For example, human computation *games* for annotating multimedia data have succeeded in collecting hundreds of thousands of tags for images [120] and music [63], a significantly larger scale than most Mechanical Turk applications (e.g., thousands to tens of thousands of tags for images [104] or speech transcription [78]).

II.C Herd It - A Social Music Annotation Game

A player arriving at Herd It (`www.HerdIt.org`) is connected with "the Herd" - all other players currently online - and the game begins. Each round of Herd It begins by playing the same piece of music to all members of the Herd. A variety of fun, simple minigames prompt players to choose from suggested tags that describe different aspects of the music they hear (Figure Figure II.2 illustrates an example of Herd It's gameplay with further examples in Section II.C.1). In every minigame, players are awarded points based on their agreement with the choices made by the rest of the Herd, encouraging players to contribute tags that are likely to achieve consensus.

Herd It's goal is to collect training data that primes and improves the machine learning system through an active learning loop by motivating human players to provide reliable descriptions of a large number of example songs using a dynamic vocabulary of tags. To achieve this goal, Herd It's development followed a *user-centered design* process [51] that aimed to create an intuitive, viral game experience. A series of rapid prototypes were released every month and tested on focus groups of 5-10 players, both in person at our lab and remotely online. During each test, we evaluated factors including playability and appeal, user-interface intuitiveness, viral potential and stability. Interviews and questionnaires completed



(a) (b)

Figure II.2 Illustration of Herd It gameplay. (a) Six bubbles float around the play area, each suggesting a mood that might be evoked by the music that is playing. The player clicks the bubble that they feel is most appropriate and all other bubbles disappear with a pop. After 15 seconds, the minigame ends and (b) the player is shown the choices made by the rest of the Herd. During this feedback period, an "agree-O-meter" fills up as other members of the Herd are revealed to agree with the player's choice. Players are awarded points equal to the percentage of the Herd that agreed with them, rewarding consensus with the Herd and implicitly collecting reliable music tags.

by test subjects were used to evaluate the extent to which players were able to focus on the music (ensuring reliable data collection), their awareness of the other players (Herd It is a social game and a player's score depends on the Herd), and overall enjoyment (indicating likelihood of large-scale participation). Feedback from test subjects led to improvements in the design and this process continued until key performance measures were reached (e.g., 90% of players said they would recommend Herd It to their friends, 100% understood the scoring system within 5 games; further results in Section II.D). Finally, wider alpha and beta releases were used to refine design features and improve performance and stability before Herd It's launch.

The user-centered design process was instrumental in determining crucial gameplay mechanics that differentiate Herd It from other music annotation games (e.g., [63, 73]). In particular our user tests discovered that, while free-text tagging works well when annotating images (which tend to feature many obvious, easily-named objects; see, e.g., [120]), many listeners found it difficult to produce and agree on a variety of tags for music in a game environment without some priming. Asking players to type their own descriptions of the music meant that the vast majority of tags were confined to a limited vocabulary of generic tags, e.g., "rock", "guitar", "drums", "male/female vocalist" (MajorMiner [73] suffers from this problem). As a result, the independent inputs of multiple players rarely converged on more interesting tags (TagATune [63] avoids this problem by asking players to guess whether they are listening to the same song, based on the free-text tags other players entered, rather than requiring agreement on the exact tags). To achieve both variety and consensus, Herd It's novel solution is to *suggest* tags for player confirmation, thereby controlling the vocabulary used to describe music while maintaining simple and compelling gameplay. In addition, tag suggestion addresses another, important design objective: it facilitates the active learning paradigm depicted in Figure Figure II.1 which requires precise control over the data collected from the Herd. Specifically, an active learning approach leverages machine learning models to suggest {song,tag} combinations that, if confirmed by human players, are most likely to produce useful training examples and optimize future model training. Herd It's tag suggestion mechanism enables this by focusing human labeling on specific {song,tag} combinations. Vice versa, Herd It's new tag suggestion design benefits from it being powered with machine intelligence. Indeed, suggesting tags randomly, rather than intelligently, was found to result in many minigames that have no relevant choices and are not fun.

Next, to achieve widespread player engagement and thus maximize training data collection, we found that Herd It should target the "casual" gamer. Unlike the traditional computer gaming demographic (i.e., teenage boys) who enjoy long-

lasting games with complicated gameplay mechanics, casual games appeal to a much wider demographic (e.g., skewed towards middle-aged women), are played in short time increments (5-20 minutes) and feature simple but addicting gameplay [57]. Herd It's simple, single-click gameplay, cartoon-ish minigame design, and intuitive scoring metric were designed to attract a broad audience of casual gamers.

Third, based on the choices offered in a given minigame, different users may end up describing a song differently, using either compatible tags (e.g., a "romantic" song that is also described as "carefree") or opposite tags (e.g., what sounds "exciting" to one listener may be "boring" to another). Given this subjectivity inherent in music appreciation, our design process revealed that it is important to evaluate agreement in minigames in a (larger) group setting, as this enables clusters of consensus to develop between the players, around multiple "right" answers. This observation inspired us to make "the Herd" a central feature of the game, rather than the player-vs-player mechanic used by other games (e.g., [63,120]). In addition, our user tests determined that *realtime*, social interaction produced more compelling gameplay than off-line group feedback (e.g., [73]). The group dynamic also makes it more difficult for a few players to cheat and gain lots of points by coordinating poor labeling (other measures to prevent cheating include randomizing tag order in minigames and preventing a single player from entering multiple games).

Finally, since many individuals use music preference to communicate information about their personality [92], players desired Herd It to be embedded in a larger social music experience. For example, they requested the ability to choose preferred genres, share music, create personal profiles and send challenges and compare scores with friends. This led us to integrate the game within the players' existing social network by publicly releasing Herd It as an application on Facebook.[8]Integrating Herd It with Facebook offers many avenues to engage players (e.g., easy login, personalized messages, player photos) and to promote the game to a wide audience (e.g., invites, challenges, fan pages). Facebook also provides

[8]`www.facebook.com`, the world's most popular social networking site with over 500 million users

demographic and psychographic information about Herd It players (e.g., gender, age, location, friend networks, favorite music), offering a hitherto unavailable level of insight into how different people experience and describe music.

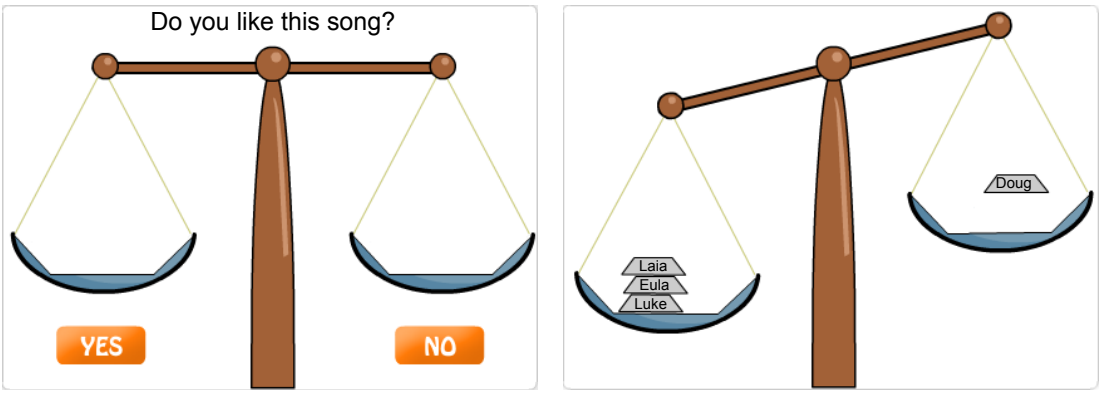
II.C.1 Herd It Minigames

In order to collect information about diverse aspects of the music as well as to enhance engagement with players, Herd It features a variety of minigames that prompt the Herd to describe the music they hear by:

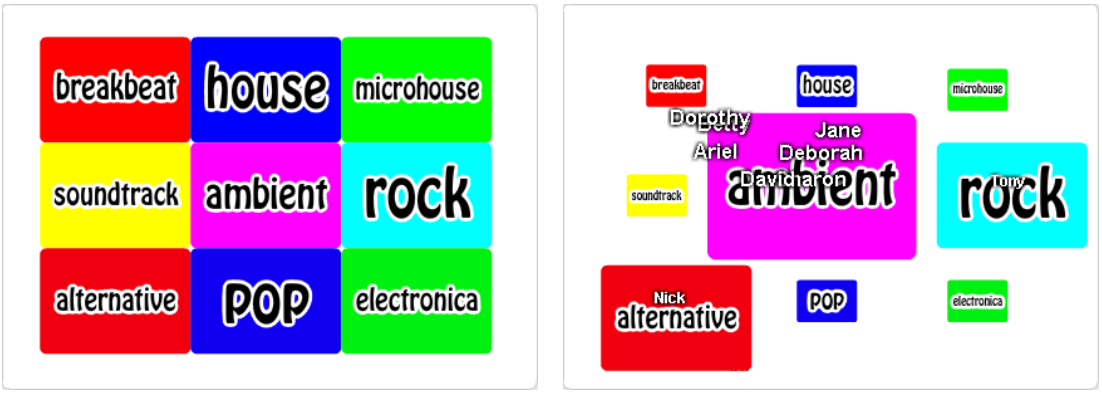
- catching floating bubbles that describe emotions or instruments present in the song,
- weighing responses to yes/no questions on a scale,
- selecting the most appropriate sub-genre from a grid,
- plotting emotional valence and arousal intensity on a Cartesian plane [59, 96] and
- choosing the color that best matches the music.

Each minigame requires a single mouse-click for players to indicate their chosen tag. In addition to the bubbles game depicted in Figure Figure II.2, screenshots in Figures Figure II.3 and Figure II.4 illustrate the remaining Herd It minigames.

Following each minigame, the player can earn 20 bonus points by correctly naming the song or the artist they have been listening to in a multiple-choice trivia round (see Figure Figure II.5 for an illustration). The sequence of one minigame and one trivia round is repeated for 5 different songs for a total of 10 rounds. At the end of 10 rounds (lasting 2-3 minutes), a summary screen presents the final scores, lists the songs that were played during the game and encourages players to connect with the Herd and the rest of their social network.

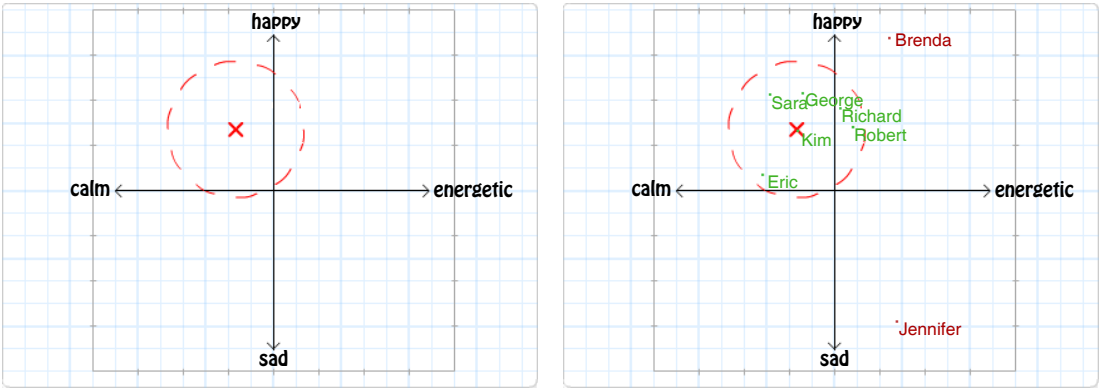


(a)



(b)

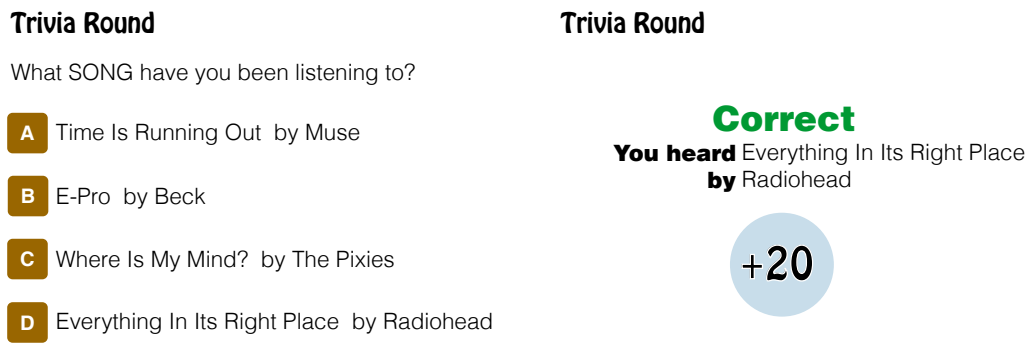
Figure II.3 Examples of Herd It's minigames that (a) weigh the Herd's response to a yes/no question and (b) determine the most appropriate sub-genre for a song.



(a)

(b)

Figure II.4 More examples of Herd It's minigames that (a) collect two-dimensional valence and arousal on a Cartesian plane and (b) enquire about the color evoked by the music.



(a) (b)

Figure II.5 Illustration of Herd It's trivia round. (a) Players can earn 20 bonus points for correctly naming the song that they hear. (b) The correct song is revealed after the player makes a choice. User testing determined that the familiar "name that tune" challenge of the trivia round encouraged novice players to participate in Herd It and the objective, "right/wrong" scoring provided a compelling counterpoint to the subjective, consensus-based scoring of the minigames.

II.D Growing the Herd

To ensure that we would generate sufficient Herd It participation to make the game-powered machine learning system viable, we engaged in a *user-centered* design process [51] to examine the effect of a variety of design features aimed at making the game fun, popular and possibly viral. Our primary goal in this formative design process was to create a core gameplay experience that was understood by the majority of players and discover problems with the interface that would prevent reliable data collection. Over a 10-month period we conducted regular user-studies both in our lab and in controlled online environments. Each test included at least 5 invited subjects and focused on *issues-based* (e.g., interviewer observed user mistakes or confusions) and *self-reported* metrics (e.g., user verbally expressed frustration or displeasure) [108]. Of the many innovative design features this iterative process inspired and tested, a handful were found to be crucial for improving interface usability and gameplay efficacy, including:

- computing player scores from percentage agreement with the rest of the Herd, rather than an arbitrary scoring metric,
- customizing feedback animations that present a player with the Herd's votes for each individual minigame, rather than a single, generic results screen for all minigames,
- including trivia rounds after each minigame, both to inform players about songs they hear and just for fun,
- integrating players' existing personal data and social network via Facebook, rather than requiring them to create a new identity on Herd It,

To quantify the impact of these design iterations, subjects completed a questionnaire at the end of each testing session that evaluated key gameplay metrics on a 3- or 5-point Likert scale. Table Table II.1 shows subject responses from the final user test, at which point we had confidence that players who tried the game would likely understand Herd It and contribute meaningful data.

Table II.1 Subject responses to final user test of user-centered gameplay design.

Are you aware of other people playing with you?	Very aware 64.3%		Somewhat 28.6%	Not at all 7.1%	
Did questions and words correspond to the music?	Clearly 100%		Sort of 0%	Confusing 0%	
Overall, how did you like the game?	Great 35.7%	Good 57.1%	OK 7.1%	Bad 0%	Awful 0%

Our secondary goal during the formative design process was to create an enjoyable, positive experience for Herd It players so as to maximize time spent playing and user uptake. In order to grow the Herd, and thereby the amount of data collected, this process inspired features that enabled players to:

- recommend Herd It, share scores and issue challenges to Facebook friends
- share music discovered during the game with Facebook friends
- chat in real-time with members of the Herd.

Figure Figure II.6 plots overall subject enjoyment of updated iterations of this user-centered game design, refined based on player input. By the end of the design process, users consistenly rated their experience as "Good" or "Great" and said they were likely to share the game with their friends or challenge friends' high scores (see Table Table II.2).

Table II.2 Social engagement metrics measured at the final user test.

	Definitely	Likely	Maybe	Unlikely	No Way
Would you try to beat a friend's high-score?	42.9%	50%	7.1%	0%	0%
Would you recommend Herd It to your friends?	44.4%	44.4%	11.1%	0%	0%

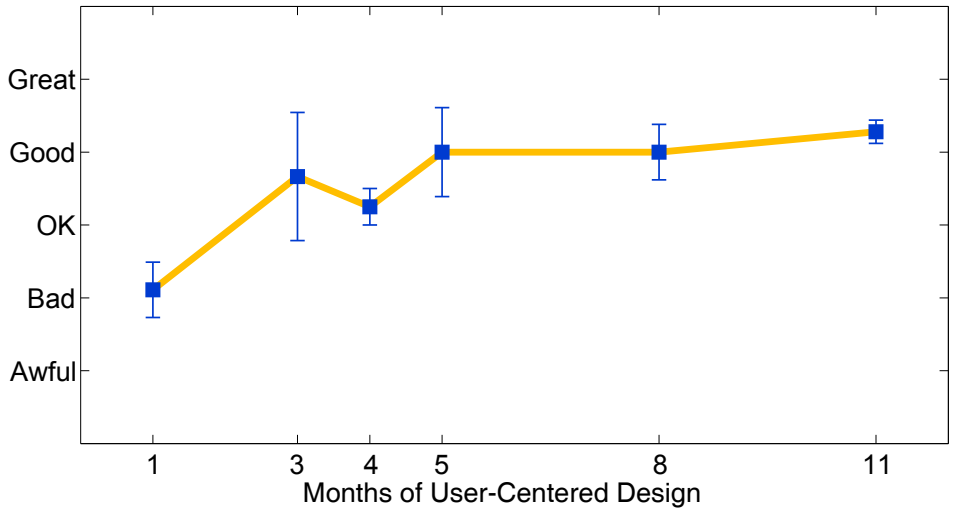


Figure II.6 Evolution of overall game enjoyment over 10 months of user-centered design and development.

In the next step of the design process, the most current version of Herd It underwent continuous, larger-scale testing. The game was exposed to 200 online users who provided feedback actively (web surveys, emails) and passively via live-site metrics [108] (e.g., ratio of new visitors who registered, clickthrough rates, number of games played, time on site). During this phase of the design process, the core gameplay experience remained unchanged. We focused on enabling and testing features designed to catalyze the continuous recruitment of human music labelers, i.e., i) acquire new users and ii) encourage existing users to return and play more games. For example, in order to track progress and save demographic details, a new player arriving at HerdIt.org was required to "add" the Facebook application before playing Herd It. Although this registration step was very simple (one button click: similar to adding a Facebook friend), it proved to be a barrier as not all players understood Herd It or why they should share their personal information. Our user tests determined that we achieved more registrations when new players were launched directly into a short demonstration game, rather than

being required to register immediately. In addition to instructing players on the rules of Herd It, this scripted demo was chosen to include well-known, popular songs and fun tags that would appeal to a wide audience and entice new users to continue playing. Once registered, a new player was brought to the Herd It home page. Multiple iterations of the home page design revealed that simplicity is key: to maximize time spent playing, the home page offers just a few simple buttons that immediately launch the user into the game. All ancillary features (high score tables and statistics, friend invites, more information about Herd It, music search, etc.) were removed to secondary pages accessed from a list of tabs.

Having enticed a new player to join the Herd, a number of design and gameplay features were included to offer deeper content that would encourage them to return and to invite their friends to join. Indeed, one of the motivations for the social elements in Herd It's design (i.e., multiple simultaneous players, group-based scoring, Facebook integration, sharing of songs and scores, etc.) was to aid in the viral distribution of the game. For example, players were prompted to invite their friends at the end of a game where they accomplished certain achievements (e.g., setting a high-score, advancing in rank or surpassing a friend's score). Increasing ranks were awarded as users scored more points (e.g., "beginner", "rock star", "hip hop hero") and a scoreboard page tracked each user's progress daily, weekly and monthly and compared to their friends. Users could post clips of the songs they had enjoyed while playing Herd It on their Facebook wall, sharing the musical experience with their friends. Finally, a blog described some of the science behind Herd It and polled users about suggested improvements to the game.

Once the design process was complete and the game was launched, Herd It was promoted to a wide audience of wouldbe players. We leveraged a number of external promotional channels, the advantages, effectiveness and drawbacks of which are summarized in Table Table II.3. including: personal emails to friends and co-workers; viral promotion through players' social networks (suggesting Facebook friends to invite, issuing high-score challenges to friends and sharing songs on

Table II.3 Pros and cons of Herd It promotional channels.

Emails to friends, family, coworkers and music research mailing lists.	**pros:** high response rate from personal acquaintances **cons:** not sustainable, responders were not "gamers" and played few games on average
Viral social network promotion: suggesting Facebook friends to invite, issuing high-score challenges to friends and sharing songs on Facebook wall.	**pros:** automatic, magnified traffic by leveraging the social network of each new user **cons:** low response rate to automated requests, pop-up requests distract players
Affiliate promotion by contacting musicians on MySpace and offering to include their music in Herd It.	**pros:** artists were eager to expose their music to potential new fans in Herd It **cons:** relied on artists to promote Herd It to their fans, slow, limited responses from most artists
Articles and interviews about music research and Herd It released in the scientific and popular press.	**pros:** wide distribution **cons:** difficult to arrange, low percentage of audience converted to players
Posting articles to blogs and online media.	**pros:** best combination of wide distribution and audience response. The single biggest source of traffic came in response to an article mentioning Herd It on the technology news site `slashdot.org` **cons:** sporadic and "bursty" – large response to each article but traffic declined sharply after 2-3 days

Facebook wall); affiliate promotion by inviting musicians to include their songs in Herd It and then promote the game to their fans; media articles and interviews, both in print and online (e.g., blogs, technology news sites).

II.E Automatic Music Tagging

Statistical pattern recognition methods for tagging music begin by extracting *features* that summarize properties of the acoustic waveforms, essentially "listening" to the musical signal. By considering a training set of reliably-labeled songs, supervised machine learning algorithms identify statistical regularities in these acoustic features that are predictive of descriptive tags like "bluegrass", "banjo", "mellow" or "slow". Machines can then generalize this knowledge by detecting the presence of similar patterns in vast catalogs of new, untagged music, thereby leveraging the accuracy of human labeling (to obtain the training set) with the scalability of automated analysis to tag this new music content (once trained). Machine learning methods for automatic music tagging continue to improve and, given training data of sufficient quality, their accuracy approaches the ceiling set by the inherent subjectivity of describing music with tags [111].

To thoroughly evaluate the efficacy of the game-powered machine learning paradigm depicted in Figure Figure II.1, we consider various state-of-the-art autotagging algorithms for its machine learning component. This includes generative [28, 111] and discriminative [38, 72] approaches. Generative methods focus on estimating the (class-conditional) distribution (e.g., with a Gaussian mixture model (GMM), dynamic texture mixtures (DTM), etc.) of acoustic features that are common among songs that human "trainers" have labeled with a given tag. By evaluating the likelihood of features from a new song under the learned distribution, the model determines the probability that the tag is a relevant description of the song [111]. Discriminative methods, on the other hand, directly optimize a decision rule to discriminate between a tag being present or absent for a given audio clip. Similarly, evaluating the decision rule for a new song allows to obtain tag probabilities (see following section for details). Just as Internet search engines rank web-pages by their relevance to a text query, the tag probabilities output by a model can be used to rank songs by their relevance to the tag.

II.F Automatic Music Tagging

Machine learning approaches to modeling the association between semantic tags and spectral patterns in a musical waveform include discriminative learning algorithms [8, 38, 72, 77, 84, 102, 125], unsupervised learning algorithms [12], and generative models [28, 53, 84, 91, 111, 114]. Of these approaches, generative models are generally better suited to handling weakly-labeled data (i.e., where songs are labeled only with the presence of some relevant tags) since they estimate audio feature distributions that naturally emerge around audio content relevant to a tag, while down-weighting irrelevant outliers. Furthermore, probabilistic rankings of relevant songs for a given query tag emerge naturally from a generative model.

One of the music autotaggers used in this work, which is also the focus of our active learning approach, is implemented using the generative machine learning model of [111], based on Gaussian mixture models (GMMs). This model gave rise to a top performing automatic music tagger in the 2008 MIREX evaluation [37]. After collecting training data (with some data collection method), the associations between a vocabulary of tags, $\mathcal{V}$, and a training song, $\mathcal{X}$, are represented as $\mathbf{y} = (y_1, ..., y_{|\mathcal{V}|})$ where $y_i > 0$ if the tag w_i has been positively associated with the audio of $\mathcal{X}$ (e.g., if the consensus of Herd It players agrees that the tag w_i is a good description for the song) and $y_i = 0$ otherwise. Figure Figure II.7(a) shows an example of a group of songs that are all described with the tag "romantic". Spectral feature vectors $\mathbf{x}_i$ extracted from the audio waveform at regular time intervals, represent a song as a collection of vectors, or "bag of features", $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$, where T is proportional to the length of the song (Figure Figure II.7(b)). The system learns a Gaussian mixture model (GMM) of the audio features for each song, using the standard expectation-maximization (EM) algorithm [32] (Figure Figure II.7(c)). These *song-level* GMMs are then combined efficiently into a *tag-level* GMM using the hierarchical EM algorithm of [115] (Figure Figure II.7(d)). The result is a model of $P(\mathbf{x}|w_i)$, the distribution of acoustic features $\mathbf{x}$ that are associated with

tag w_i:

$$P(\mathbf{x}|w_i) = \sum_{n=1}^{N} \alpha_n^i \mathcal{G}(\mathbf{x}|\mu_n^i, \Sigma_n^i),$$

where $\mathcal{G}(\cdot|\mu, \Sigma)$ is a multivariate Gaussian distribution with mean μ and covariance Σ, and the mixture weights α_n^i are such that $\alpha_n^i \geq 0, \ \forall n$ and $\sum_{n=1}^{N} \alpha_n^i = 1$. In this work, we use $N = 16$ component GMMs to model each tag.

To label a new song $\mathcal{X}$ using the vocabulary of tags, modeled as above, the likelihood of the bag of features that represents the entire song is inferred under the learned tag-level models using the naïve Bayes assumption of independence between features: $P(\mathcal{X}|w_i) = \prod_{t=1}^{T} P(\mathbf{x}_t|w_i)$ (Figure Figure II.8(c)). Posterior probabilities of each tag, for the new song $\mathcal{X}$, are found using Bayes' rule:

$$P(w_i|\mathcal{X}) = \frac{P(\mathcal{X}|w_i)P(w_i)}{P(\mathcal{X})},$$

where $P(w_i)$ is the prior probability that tag w_i will appear in an annotation and is assumed to be uniform; $P(w_i) = 1/|\mathcal{V}|$. The song prior, $P(\mathcal{X})$, is obtained by summing the song likelihoods over all $|\mathcal{V}|$ tags in the vocabulary:

$$P(\mathcal{X}) = \sum_{v=1}^{|\mathcal{V}|} P(\mathcal{X}|w_v)P(w_v).$$

The final result is a set of *semantic weights*, $P(w_i|\mathcal{X}), \ \forall w_i \in \mathcal{V}$, probabilities that suggest how well each tag in the vocabulary describes the song's acoustic content. The semantic weights for each tag are collected in a *semantic multinomial*, a probability distribution that provides a rich description of the acoustic content of a song (Figure Figure II.8(d)). While the alternative autotagging algorithms examined in this chapter (i.e., [28, 38, 72]) use different models of the acoustic content associated with each tag, they each allow to compute a similar probabilistic description of the semantics of a song's content. Given a semantic query, based on a tag or set of tags, the relevant dimensions of the semantic multinomials are selected to automatically rank songs by their relevance to the query.

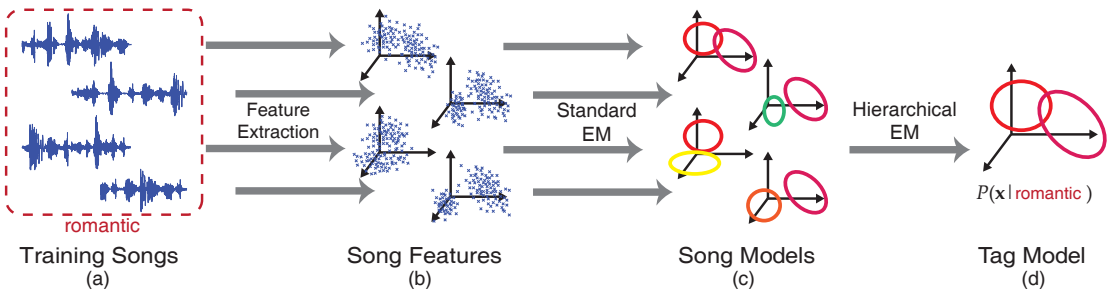


Figure II.7 Training an automatic music tagger. (a) Human labelers provide a training set of songs that have been reliably labeled with a given tag (e.g., "romantic"). (b) A "bag of features" represents the waveform of each training song. (c) A Gaussian mixture model (GMM) of the acoustic features of each song is learned using the EM algorithm [32]. (d) The GMMs for each song are combined efficiently into a single GMM that models the acoustic features predictive of the tag using a hierarchical EM algorithm [115].

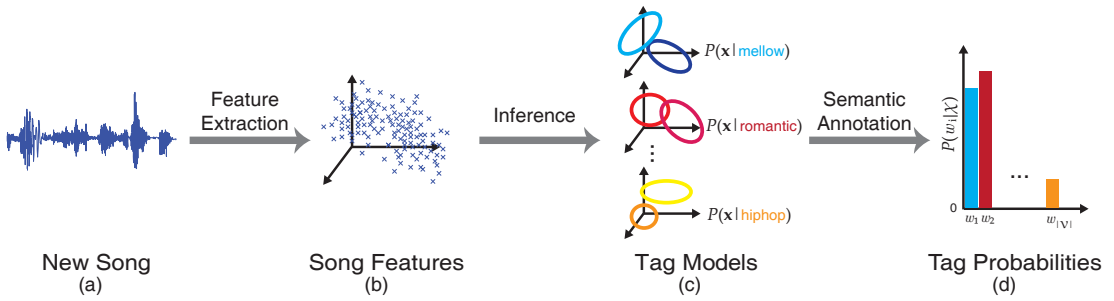


Figure II.8 Automatically tagging a new song. (a) A new, unlabeled song is to be analyzed by the automatic tagging system. (b) A "bag of features" represents the waveform of the song. (c) The features are compared to previously-learned models of each tag. (d) Tag probabilities are obtained, providing a semantic profile that describes the song's acoustic content.

II.F.1 Active Learning

Traditional machine learning approaches use a single, fixed training set to learn models that, once trained, remain static. In our game-powered machine learning framework however, new data is constantly being contributed by Herd It players. That data can be used to update our tag models. Even more, since Herd It's design permits actively focusing players' efforts on specific songs and tags, it is possible to collect specifically that data that is expected to improve tag models most effectively. This is achieved through an *active learning* approach [99], that leverages the current tag models to identify the most effective song-tag pairs for future model updates. Active learning fully integrates the autotagging algorithm with the data collection process, to optimize model training. To investigate the benefits of deploying Herd It in an active learning loop compared to updating models with randomly collected data, we develop a novel active learning algorithm to suggest data for training generative models and apply it with the GMM-based autotagger, a top performing algorithm in the 2008 MIREX (Music Information Retrieval Evaluation eXchange) evaluation of automatic music taggers [37].

Various active learning algorithms have been proposed for *discriminative* machine learning methods,[9] where both positive and negative examples are used to learn a decision boundary between classes. *Generative* approaches, on the other hand, require only positively-labeled examples for training (i.e., songs that exemplify a certain tag) and negatively-labeled training examples offer no improvement to the model.[10] To collect positively-labeled training examples and improve a generative model through active learning may suggest sampling unlabeled examples that have high likelihood under the current model and procuring labels for them (e.g., by presenting {song,tag} pairs in Herd It minigames). However, this "certainty"

[9]Strategies for actively learning discriminative models include *uncertainty sampling* [67] — where points are chosen that are least certain (or have highest entropy), under the current model (e.g., points closest to the decision boundary) — and *variance reduction* [25] — where samples are chosen to reduce the model's output variance.

[10]For example, for generative models, uncertainty sampling faces the problem that unlabeled songs which have low certainty under the current model are likely to result in negative labels.

sampling approach suffers from two drawbacks: early in training, when the model is not yet well learned, the most likely samples may not in fact be positive examples and thus will not contribute to the training set. Later in the learning process, sampling from the most likely areas results in many confirmed positive examples that conform to the model's current training set and lack the diversity required to generalize the current model to uncertain areas of the feature space. Exploration of these uncertain areas advocates for a more random sampling of unlabeled examples. Rather than a complete random sampling, we can actively increase the efficiency of the data collection and, thus, the learning rate, by reducing the likelihood of sampling unlabeled examples that are eventually labeled as negatives (which are of no use to train the generative model). We achieve this by avoiding points that most disagree with the current model. More specifically, we rank all of the unlabeled examples by their likelihood under the current model, remove the 10% of examples with lowest likelihood and query labels randomly from the remaining 90% of examples. By removing the least likely points and sampling randomly elsewhere, we aim to avoid querying labels for negative examples and achieve rapid confirmation of a diverse training set for our generative model. Preliminary experiments for GMM modeling of music tags have shown that removing the 10% least likely songs finds a good balance between exploring areas of the feature space that are poorly modeled while avoiding points that are unlikely to result in positive examples.[11]

II.F.2 Training Data Requirements

To determine the total amount of human labeling effort required to learn a reliable tag model, we evaluate the machine learning performance, given a varying number of training examples. Figure Figure II.9 shows the per-tag and average

[11] In a feature space of high dimension, d, the probability density of a Gaussian distribution with variance σ is focused on a small shell a distance $\sigma\sqrt{d}$ from the mean and thus the majority of points tend to have very similar GMM likelihoods [31]. While this fact can make it difficult to identify positive points based on likelihood, any points that have significantly *lower* than average likelihood can be excluded with confidence.

performance of the GMM-based autotagger when trained on {10, 25, 50, 100, 133} *CAL500* songs per tag. Although absolute performance varies between individual tags, on average, autotagger performance plateaus with 100 training examples. With 2 clicks required to confirm one {song,tag} association and 100 songs sufficient to learn a reliable tag model, our game-powered machine learning system requires approximately 200 human inputs to capture the essential information about a tag.

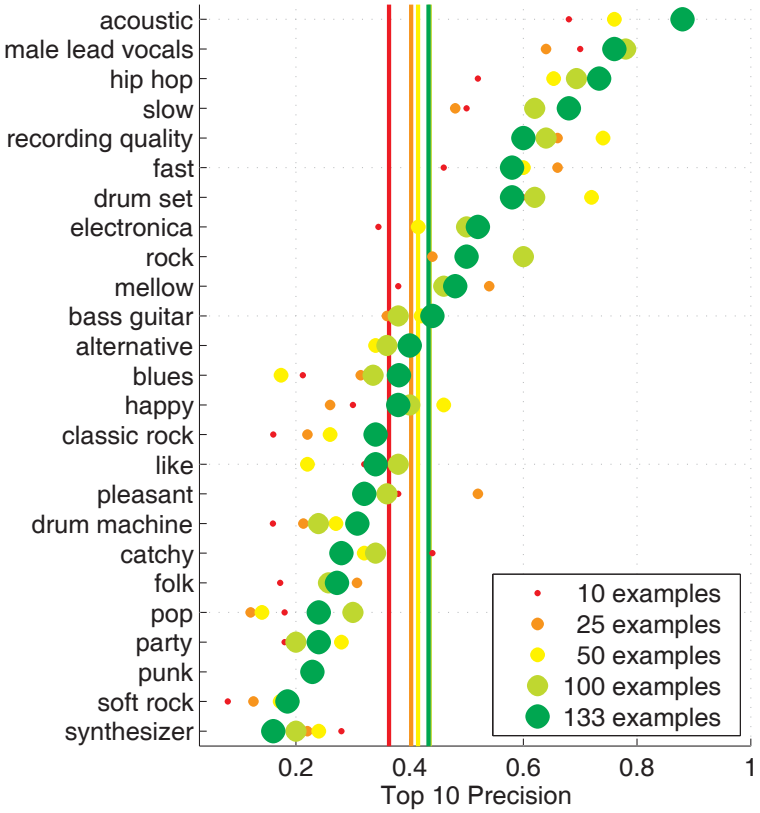


Figure II.9 Autotagging top-ten precision as a function of the number of training examples. For 25 tags in the *CAL500* dataset with at least 133 songs, we train multiple autotagging models by limiting the number of training examples to {10, 25, 50, 100, 133} songs. Vertical lines show the average performance at each limit: 10 songs is clearly insufficient but no improvement is gained beyond 100 training examples (the averages for 100 and 133 examples are overlapping).

II.G Game-Powered Machine Learning

Our game-powered machine learning approach aims to collect sufficient human labels, through game play, to train an automatic music annotation system that can reliably generalize semantics to unlimited music. Qualitatively, we argue that this crowdsourced approach is superior to requiring expert annotators as it is less costly, more scalable and collects a dynamic dataset that can be adapted over time to focus on the most relevant or important tags. To quantify the efficacy of our game-powered machine learning framework, we conduct experiments designed to answer the following two questions: i) Can machine learning algorithms be trained with data collected from Herd It's crowd of non-experts as accurately as with data collected from paid expert musicologists? ii) Can accurate tag models be learned with less human effort by encapsulating Herd It in an active learning framework? To answer these questions, we deployed Herd It online, engaging 7,947 people to provide over 140,000 clicks that associate songs with tags through 5 different types of minigames.

To generate minigames in a passive system, without active learning, we first generate a list of 10-20 candidate {song,tag} pairs, chosen randomly from the authors' personal music collection of over 6,000 popular songs from the past 70 years, and a vocabulary of 1,269 tags, including sub-genres, emotions, instruments, usages, colors and more. To generate a particular minigame, one {song,tag} pair is selected from the list of candidate pairs, biased by associations[12] with the musical genre Herd It players selected before starting the game (pop, rock, hip-hop, blues, electronica or "everything") and by the specific minigame (e.g., certain minigames focus on sub-genres, colors or bi-polar adjectives). The remaining tags for the minigame (each minigame suggests between 1 and 9 tags for player confirmation) are chosen from the same tag category. Candidate {song,tag} pairs remain on the list until they have been viewed in minigames by a maximum of 50 players. At that point, a candidate {song,tag} pair is removed from the list and replaced by

[12] determined using the online music service `http://last.fm`

a new, randomly sampled one. Maintaining a reasonable list of candidate pairs ensures diverse game play.

Consensus between players' clicks collected in Herd It minigames is used to "confirm" reliable {song,tag} associations. More specifically, the generative model of labels, accuracies and difficulties, or "GLAD", [123] conceives of each human input as an estimate of the underlying true label that has been corrupted by player inaccuracy and the difficulty of labeling the song. Using an expectation-maximization algorithm, GLAD optimally combines the votes from all Herd It players and we confirm the findings of [123] that the resulting consensus is more reliable than heuristics such as majority vote, percentage agreement or vote thresholds. A {song,tag} pair is presented in Herd It minigames until GLAD "confirms" a reliable association, based on the historical click data for that {song,tag} pair. If a {song,tag} pair remains unconfirmed after being viewed by 50 Herd It players, it is "rejected" and not sampled further. Overall, GLAD confirmed 8,784 {song,tag} pairs, representing song examples of 549 tags, while 256,000 pairs were rejected. To ensure that we have enough data to train robust machine learning models and answer the first question, we reduce the dataset to the 127 tags for which Herd It has identified at least 10 reliable example songs. Data was collected passively (i.e., no active learning) and this provides the baseline against which to compare an *active* learning strategy and evaluate the second question.

To answer the first question, we quantify the efficacy of our game-powered machine learning framework and compare it to "expert-trained" machine learning. That is, we evaluate the performance of a music autotagging algorithm when trained on i) the Herd It *game* data and ii) data derived from *expert* musicologists at Pandora.com's "Music Genome Project" (*MGP*), respectively. After training, the accuracy of each autotagger is evaluated on *CAL500*: an independent evaluation set of 500 songs fully labeled by multiple humans using a controlled survey [111] (see Section II.H for details about the *CAL500* and *MGP* datasets). For this comparison, we train and evaluate models of all tags that are available in both

the *MGP* and *CAL500* vocabulary and for which Herd It has collected at least ten confirmed example songs. This results in 25 tags. The models of each of these tags are used to retrieve the 10 most relevant songs from the *CAL500* corpus for each single-tag query. These top-ten search results – automatically retrieved by a machine – are evaluated by comparing to the *CAL500 ground-truth*, and computing the precision (i.e., the number of songs in the machine-ranked top-ten that the ground truth effectively associates with the tag). Finally, the precision is averaged over all 25 tags. Since both Herd It and *MGP* models are instances of the *same* machine learning algorithm, but trained on *different* data sets, any significant differences in autotagging performance most likely reflect differences in the quality of the respective training data sources. This allows us to evaluate human computation games — Herd It, in particular — as a source of reliable training data. To prevent bias induced by a particular choice of machine learning algorithm, this comparison is repeated for multiple state-of-the-art autotagging algorithms.

In addition to showing that game-powered machine learning can be competitive with an expert-trained system, in a second step, we demonstrate the efficacy of *actively* integrating machine learning with game-based data collection. The baseline here is the passive approach outlined above, which "analyzes" (i.e., confirms or rejects through human labeling) {song,tag} pairs in random order. As more {song,tag} pairs are analyzed (i.e., more human effort contributed), a tag's training set grows, tag models are updated and autotagging performance is expected to improve. We compare this to an active learning paradigm which aims to improve tag models more effectively by leveraging current models to select the next {song,tag} pairs that will be analyzed. More precisely, for each of the 25 Herd It tags that were evaluated earlier, we collect all songs that appeared with the tag (confirmed or rejected) in previous Herd It minigames. We then estimate 25 GMM-based tag models by engaging in an iterative training procedure, for each tag, based on this list of "candidate" songs. At each iteration, we first compute the

likelihood, under the current tag model, of all remaining candidate songs and use our active learning method for generative models to prioritize 10 candidate songs for analysis with that tag (for the first iteration, candidate songs are chosen randomly). That is, we use our active learning algorithm to re-sample {song,tag} pairs that were previously presented in Herd It games. Songs for which the {song,tag} pair was previously confirmed are added to the tag's training set; the remaining, rejected songs are removed from future candidate lists. Finally, we retrain the tag model using the updated training set and evaluate its performance on the *CAL500* test set. We once again recompute the likelihood of all remaining candidate songs under the updated model, actively select 10 candidate songs for analysis, retrain the tag models, and so on. This is repeated up to 200 times, analyzing up to 2,000 songs for each tag. At each iteration, we evaluate and average the performance of the 25 updated tag models. We compare this to the passive baseline, which corresponds to sampling 10 songs randomly, for each tag, at each iteration.

II.H Music Data

In this section, we describe in more detail the data used in our experiments, including the audio features, the *MGP* data and the *CAL500* data.

II.H.1 Audio Features

The method in [111] used Mel-frequency cepstral coefficients (MFCCs) [69] to capture the spectral content of short-time segments (approximately 5ms) from each song. For the GMM autotagger used in this work, we instead use the timbre coefficients computed using the feature extraction application programming interface (API) offered online by EchoNest.com, and described at http://developer.echonest.com/docs/method/get_segments/. This open API produces audio descriptors very similar in content to MFCCs but combines feature values over longer-time windows of homogenous audio (variable length but approxi-

mately 250ms), resulting in a more concise representation of each song (i.e., 100's vs. 10,000's of feature vectors per song). Tingle et al. recently showed that these EchoNest *timbre* feature vectors outperform MFCC feature vectors on the task of automatic music tagging when using the GMM-based system [106]. As a result, we use this feature representation and similarly find a (slight) improvement in performance (results not shown). For the machine learning methods that represent a song as a single feature vector (e.g., SVM [72] and Boosting [38]), we follow [72] and represent each song as the concatenated mean and variance of it's EchoNest timbre features. For the DTM model, which requires a constant interval between extracted feature vectors, we use the MFCCs described above.

Note: To augment *content-based* music search (based on features that describe the acoustic content only), a wealth of *metadata* (e.g., artist and song names, web search results [60], lyrics, song reviews, artist biographies, chart position, playlists, etc.) can be collected. To take advantage of this additional information to augment music search, one possible direction that has been explored, for example, is the use of kernel-based methods that can learn from multiple kernels. Such approaches combine kernels derived from the acoustic waveform with kernels derived from metadata extracted online, when available. This improves the performance of using either data source in isolation (e.g., [8, 74]). However, while metadata is readily available for popular songs, in the case of undiscovered songs, where a music search engine would be most useful, this information can not be relied upon (e.g., new songs have not yet been reviewed, unknown artists do not have lyrics or biographies available online). Thus, to satisfy the most general use case (including undiscovered music), our game-powered machine learning solution focuses solely on the only data that is guaranteed to be available: the acoustic waveform.

II.H.2 Training on Expert Labels: the Music Genome Project

The Music Genome Project (*MGP*), a subsidiary of the Internet radio station `Pandora.com`, employs musicological experts to annotate music using com-

prehensive surveys. Over the past ten years and at a cost of many millions of dollars, the *MGP* has labeled hundreds of thousands of songs with up to 500 tags.[13] While these labels are presumably of very high quality, the *MGP* lexicon is static: adding new styles of music or translating to another language requires laboriously re-tagging all songs with the new vocabulary and, as musical styles change, the tags can not be updated (e.g., contrast tags used to describe "classical" music with those relating to "hip hop" or even "hip hop" in 1981 with "hip hop" in 2011). Although we do not have access to the complete *MGP* data, we have collected a few *MGP* tags for 10,000 songs by retrieving publicly-available information displayed on `Pandora.com` [106]. Training machine learning models on this *MGP* subset allows to evaluate our music retrieval system when trained on data derived from a small group of expert labelers.

II.H.3 Evaluating on *CAL500*

At UCSD's Computer Audition Laboratory (CAL), we collected ground-truth data, similar to the Music Genome Project, by paying undergraduate music students to listen to songs and complete a survey that labeled the songs with relevant tags. The *CAL500* dataset consists of 500 songs annotated with 149 tags from categories related to musical genres, emotional content, instrumentation, vocal characteristics and activities during which one might listen to the song [111]. Each song was annotated by at least three individuals. It is important to note that, unlike most data mined from online sources, the *CAL500* dataset includes both positive and negative associations between songs and tags. This makes *CAL500* suitable for *evaluating* the output of automatic tagging systems as it is possible to judge correct — and also *incorrect* — results[14].

The *CAL500* songs were used only for testing the machine learning models; none were included in the training sets. Each learned model is used to rank all the

[13] `http://blog.pandora.com/faq/`

[14] Datasets that do not explicitly label negative associations between tags and songs are *weakly labeled*; the absence of a label may mean that the tag is truly not relevant or that no data was available for this song-tag pair.

CAL500 songs by their relevance to the tag. For example, the model for the tag "jazz" orders songs by how well they match patterns common to the jazz music in the training set. The ranking is evaluated in reference to the *CAL500* ground-truth labels by computing the *top-ten precision*, the proportion of relevant songs among the first ten results (e.g., the true number of "jazz" songs in the top-ten)[15]. High top-ten precision means that many songs at the top of the machine-ranked list are appropriate results for the query tag — exactly the goal of a music search engine[16].

II.H.4 Tag Vocabulary

To learn reliable models, we consider only the 127 tags for which Herd It players have confirmed at least ten training examples. Of these, we limit our evaluation to the 25 tags that also appear in both the *MGP* dataset and the *CAL500* ground-truth dataset. Table Table II.5 details the size of the resulting vocabulary used in our evaluation.

Table II.4 Herd It tags collected. Although Herd It has collected data to train machine learning models of 127 tags, only the 25 tags that are also found in the *CAL500* and *MGP* datasets are used for evaluation.

1,269	unique tags used by Herd It players
549	... confirmed by GLAD algorithm [123]
127	... with at least 10 example songs confirmed
25	... overlap with *CAL500* and *MGP* datasets

[15] If there are only $n < 10$ relevant songs in the ground truth, only the top n results are evaluated.

[16] Other metrics that evaluate the complete ranking, such as mean average precision or the area under the receiver operating characteristic curve, exhibit qualitatively similar trends to the results reported here.

II.H.5 Dataset Availability

The song labels from all datasets collected for this work – Herd It, *MGP* and *CAL500* – are available on the Computer Audition Lab webpage `http://cosmal.ucsd.edu/cal/`. Each dataset is presented as a flat text file where each line has the format:

`<song-name><TAB><tag-name>`

While copyright issues preclude distribution of the audio files used in this work, all audio features are made available.

II.I Results

II.I.1 Experiment 1: Comparison to Expert Annotations

Table Table II.5 presents the average precision of the top-ten music search results for 25 single-tag queries on *CAL500*, achieved by training four state-of-the-art autotagging algorithms on Herd It's data. This is compared to the performance obtained by training on expert *MGP* data. For each of the 25 tags common to Herd It, *MGP* and *CAL500*, we evaluate the top-ten precision on *CAL500* and average performance over all tags. While the absolute performance depends on the machine learning method used, the *relative* performance between models trained using Herd It and those that use *MGP* data remains consistently over 95%. These findings answer our first question by demonstrating that a game-based machine learning system, trained on data collected from Herd It players, provides a competitive alternative to a system trained on expert labeled data, across a variety of algorithms.

Figure Figure II.10 offers a more detailed comparison of a Herd It and a *MGP* based system, by examining the performance of each tag model learned by the hierarchical GMM algorithm [111]. The ability of the machine learning algorithm to model different tags varies (explaining the relatively large standard errors in Table Table II.5), e.g., "acoustic", "male lead vocals" and "hip hop" songs are more

Table II.5 Average top-ten precision of four autotagging algorithms trained on Herd It examples and tested on the *CAL500* dataset; also shown is the relative (top-ten precision) performance of these Herd It-trained models compared to models trained on expert *MGP* examples. "Random" shows the expected performance from random guessing.

Autotagging algorithm	Top-10 Precision Herd It training	Herd It vs. *MGP*
Hierarch. GMM [111]	0.40	95.8% ±4.6
Hierarch. DTM [28]	0.42	98.9% ±6.0
Boosting [38]	0.38	99.7% ±4.2
SVM [72]	0.38	95.6% ±7.2
Random	0.18	-

easily identified, while "hand drums" and "funk" music are poorly modeled. In general, model performance is independent of the training data source (i.e., most points lie close to the diagonal in Figure Figure II.10, indicating comparable results for each system). Only the tag "synthesizer" was modeled significantly better using *MGP* training data while the model trained on Herd It's examples of the tag "drum set" was significantly better (2-tailed t-test, 95% significance level). In summary, Table Table II.5 and Figure Figure II.10 quantitatively demonstrate that training from Herd It's crowdsourced data captures knowledge similar to training from expert annotations.

II.I.2 Experiment 2: Active Learning

We turn now to the second question: can integrating machine learning and Herd It's game-powered data collection in an active learning loop train accurate models with less human effort than a passive system? To measure human effort,

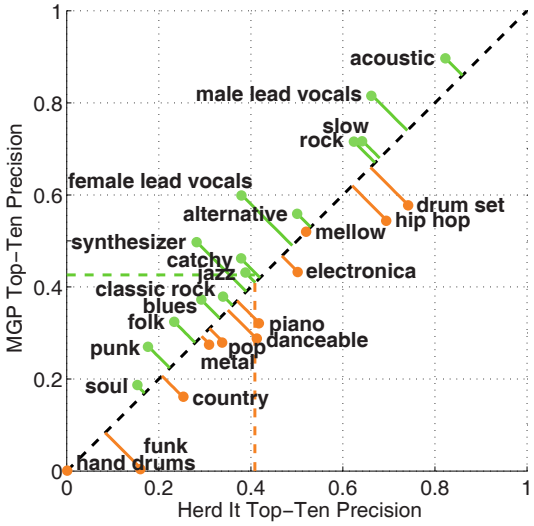


Figure II.10 Top-ten precision for machine learning models trained on Herd It's crowdsourced data (x-axis) and models trained on data from the Music Genome Project (y-axis). While absolute performance depends on the tag (e.g., "acoustic" music is better modeled than "soul" music), on average (dashed lines) Herd It's crowdsourced data trains models that are as precise, at the tag-level, as models learned from expert-labeled data.

we consider the number of {song,tag} pairs analyzed through Herd It gameplay, expressed as the number of songs analyzed per tag (for each of the 25 tags being modeled). Figure Figure II.11 displays the improvement in song retrieval performance of the GMM autotagging algorithm as more songs are analyzed for each tag (and, consequently, more training examples collected for model estimation), following both an active learning and a random sampling strategy. The results demonstrate an improved learning rate due to active learning: active learning requires analyzing, on average, 550 songs per tag to get within one standard error of expert-trained performance, while the passive strategy hits this performance level after analyzing 1090 songs for each tag. We see significantly improved performance due to active learning between 520 and 600 analyzed songs (paired t-test, p=0.05). Fig. 4 highlights the improved efficiency by shading the learning curves while performance is outside the *MGP* error: by prioritizing the order in which

{song,tag} pairs are presented to players, our active learning approach reduces the human labeling effort required by half. A more detailed inspection of the results reveals that active learning confirms an average of 36 training songs per tag, out of 550 analyzed candidates, versus 54, out of 1090, for random sampling. This shows that active learning boosts the learning rate by suggesting fewer {song,tag} pairs that eventually get rejected and are of no use for training (compared to suggesting random {song,tag} pairs) while still producing a sufficiently diverse set of confirmed training songs.

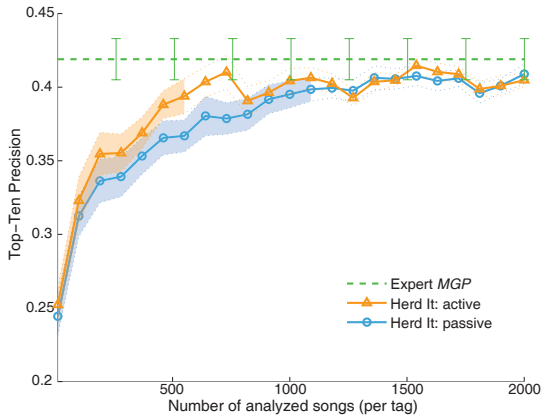


Figure II.11 Music autotagging performance as a function of human effort (i.e., number of songs analyzed by Herd It players). For each of the 25 tags considered, "passive" randomly selects songs for analysis while "active" leverages an active learning paradigm. Each {song,tag} pair appeared in Herd It minigames until it was either confirmed by GLAD or rejected after being presented to 50 players. Y-axis plots the average precision of the top-ten search results returned by tag models trained on all songs confirmed by Herd It at each 100-song increment on the x-axis. Bands show the standard error of the mean and remain shaded while performance is inferior to expert-trained models (550 songs for active, 1090 songs for passive). Integrating active machine learning with Herd It's data collection improves the learning rate, achieving target performance (within error bands) with less human effort.

Having demonstrated that Herd It data can train automatic music taggers

that are as accurate as an expert-trained system, we now compare the efficiency of collecting tags from amateur players with that from trained experts. Pandora's musicological experts take 20-30 minutes to analyze and quantify the association between a song and 100-500 semantic dimensions,[17] a rate of about 12 song-tag associations per expert-minute. Herd It minigames last about 30 seconds and present, on average, 5.4 tags for player analysis. Thus, a single Herd It player analyzes 10.8 song-tag associations per minute, a little less than the Pandora expert. To quantify (i.e., confirm or reject) a song-tag association, the analysis of up to 50 players is required, versus that of one Pandora expert. So, Herd It's game-based approach gathers reliable tags from humans for free with about 2% the efficiency of paid, expert labeling. Of course, Herd It's lower efficiency is multiplied by the number of simultaneous players in the Herd, which could be significantly larger than the number of musicological experts that can be gainfully employed, simultaneously.

Moreover, while a human-only approach requires the same amount of labeling effort for the first song as for the millionth, our game-powered machine learning solution needs only a small, reliable training set before all future examples can be labeled automatically. The latter improves efficiency and cost by orders of magnitude. Tagging a new song takes about 4 seconds on a modern CPU: with 8 parallel processors, it would take just a week to tag 1 million songs or annotate Pandora's complete song collection, which required a decade of effort from dozens of trained musicologists. Table Table II.7 presents results for 7 example songs, randomly chosen from personal music collections and never presented to the system before. Each song is analyzed using the GMM autotagger trained on all 127 tags for which Herd It has collected at least 10 reliable examples and then described by inserting the most likely genre, instrument, emotion, color and usage tags into a template sentence (as in [111]). The resulting "robot reviews" qualitatively illustrate that the machine learning models trained on Herd It data reliably label new music with a variety of tags. In general, musically objective

[17] http://blog.pandora.com/faq/

tags (e.g., "hip hop" and "disco") are well-modeled by machine learning while the subjective tags collected by Herd It's more whimsical minigames are harder for machine learning models to predict (e.g., the color evoked by the music or songs that are "atmospheric" or "sexy").

II.I.3 Experiment 3: Comparison to Other Music Annotation Games

The first principal experiment, comparing systems based on Herd It versus *MGP* data, demonstrated that consensus derived from multiple non-experts playing a human annotation game can train a multimedia retrieval system as reliably as data from expert labelers. With Herd It, we designed a new, custom solution as opposed to using existing music annotation games such as TagATune [63] or MajorMiner [73] as a source of training data. In particular, TagATune connects pairs of players and asks them to describe the music they hear by typing tags on their computer keyboard. Based on their partner's tags, players must guess if they are listening to the same song or not. Likewise, MajorMiner's tagging is entirely user-driven. Herd It, on the other hand, is based on tag suggestion (and confirmation by users), to focus human effort on a larger, more varied vocabulary than TagATune or MajorMiner — which tend to focus on a small set of short, obvious words like, e.g., "drums", "rock", "guitar", etc. Moreover, while the aim of previous games was to have music directly labeled by humans, we designed Herd It explicitly to collect training data for labeling music with machine learning. By suggesting tags for players to confirm, rather than asking players to input tags, Herd It allows to focus data collection on tags (and songs) that are of most interest to the modeling process, in a dynamic way (enabling an active learning loop, as opposed to TagATune's or MajorMiner's more open-ended data collection).

To validate the motivation for designing Herd It as a new game, we empirically compare GMM-based models trained on TagATune and Herd It data. TagATune has released[18] a dataset similar to Herd It's: 10,000 song clips labeled

[18] `http://tagatune.org/Magnatagatune.html`

Table II.6 Comparison of the average number of training examples available from various data sources and the resulting music tagging performance. Top-ten precision, averaged over the 14 tags in common between all three data sources and *CAL500*, is evaluated in reference to the *CAL500* ground-truth, after training GMM-based models for each tag. While collecting the fewest number of songs for each tag, Herd It clearly provides more reliable examples for training machine learning models than TagATune. Models trained on examples collected by Herd It perform significantly better than those learned from TagATune data (paired t-test, 95% significance level).

	Songs per Tag	Top-Ten Prec.
TagATune [63]	88	0.368
MGP [106]	851	0.422
Herd It	46	0.424

with 188 user-generated tags. MajorMiner data was not explored as it is a private evaluation set for the MIREX challenge [37]. Evaluation of GMM-based autotagging models of the 14 tags that are common to TagATune, Herd It, *MGP* and *CAL500* demonstrates that, even with fewer examples of each tag, groups of Herd It players provide more reliable training data than do pairs of TagATune players (see Table Table II.6).

II.J Conclusions

We proposed game-powered machine learning as an integrated, scalable, affordable and reliable solution to powering semantic search of massive amounts of multimedia content, and investigated its efficacy for music search. Herd It, an online music annotation game, collects reliable examples of how humans use semantic tags to describe music. Although this human computation approach is insufficient to

label the millions of songs available on the web, the knowledge collected by our game trains machine learning algorithms that can generalize tags to vast amounts of new, unlabeled music. Compared to other music games with a purpose, Herd It was specifically designed for and is actively integrated with the machine learning algorithms, to provide the data that most effectively trains them.

Our results demonstrate, first of all, that game-powered machine learning is as good as expert-based machine learning – i.e., annotations collected from human computation games allow to train autotagging models as accurately as expensive, expert annotations – while offering some distinct advantages (e.g., cost-effectiveness, scalability, flexibility to update tags presented in the game, or replace them to focus learning on different tags of interest). Second, we show that embedding Herd It in an active learning paradigm allows to train accurate autotaggers more effectively, i.e., with less human effort, compared to a passive, feed-forward approach. We conclude that actively integrating human computation games and machine learning – combining targeted data collection by annotation games, directed by machine learning, with automatic prediction by scalable machine learning algorithms – enables simple, widespread multimedia search and discovery.

II.K Acknowledgements

Chapter 2, in full, is a reprint of the material as submitted for publication in the Proceedings of the National Academy of Science. L. Barrington, D. Turnbull and G.R.G Lanckriet, 2011. The dissertation author was the primary investigator and author of this paper.

Table II.7 Automatic music summaries produced by GMM-based machine learning models trained on Herd It data. For each song, the tags in **bold** are automatically determined by the automatic tagging system to be the most appropriate genre, instrument, emotion, color and time.

Wham! "Careless whisper"	This **soft rock** song features **male lead vocals**, feels **mellow**, evokes the color **white** and would be good to listen to **on a rainy day**.
Neil Young "Heart of gold"	This **folk** song features **bass guitar**, feels **slow**, evokes the color **orange** and would be good to listen to **late at night**.
Michael Jackson "The way you make me feel"	This **disco** song features **drum set**, feels **catchy**, evokes the color **yellow** and would be good to listen to **at dusk**.
Metallica "One"	This **rock** song features **male lead vocals**, feels **atmospheric**, evokes the color **orange** and would be good to listen to **in the morning**.
Lady Gaga "Poker face"	This **hip hop** song features **drum set**, feels **happy**, evokes the color **red** and would be good to listen to **at a party**.
The Flying Burrito Brothers "White line fever"	This **folk-rock** song features **piano**, feels **acoustic**, evokes the color **orange** and would be good to listen to **in the morning**.
Eminem "Kill you"	This **hip hop** song features **drum machine**, feels **sexy**, evokes the color **black** and would be good to listen to **late at night**.

Chapter III

Semantic Annotation and Retrieval of Music and Sound Effects

III.A Abstract

We present a computer audition system that can both *annotate* novel audio tracks with semantically meaningful words and *retrieve* relevant tracks from a database of unlabeled audio content given a text-based query. We consider the related tasks of content-based audio annotation and retrieval as one supervised multi-class, multi-label problem in which we model the joint probability of acoustic features and words. We collect a data set of 1700 human-generated annotations that describe 500 Western popular music tracks. For each word in a vocabulary, we use this data to train a Gaussian mixture model (GMM) over an audio feature space. We estimate the parameters of the model using the *weighted mixture hierarchies expectation maximization* algorithm. This algorithm is more scalable to large data sets and produces better density estimates than standard parameter estimation techniques. The quality of the music annotations produced by our system is comparable with the performance of humans on the same task. Our 'query-by-text' system can retrieve appropriate songs for a large number of musically relevant words. We also show that our audition system is general by learning a model that can annotate and retrieve sound effects.

III.B Introduction

Music is a form of communication that can represent human emotions, personal style, geographic origins, spiritual foundations, social conditions, and other aspects of humanity. Listeners naturally use words in an attempt to describe what they hear even though two listeners may use drastically different words when describing the same piece of music. However, words related to some aspects of the audio content, such as instrumentation and genre, may be largely agreed upon by a majority of listeners. This agreement suggests that it is possible to create a computer audition system that can learn the relationship between audio content and words. In this paper, we describe such a system and show that it can both

annotate novel audio content with semantically meaningful words and *retrieve* relevant audio tracks from a database of unannotated tracks given a text-based query.

We view the related tasks of semantic annotation and retrieval of audio as one supervised multi-class, multi-label learning problem. We learn a joint probabilistic model of audio content and words using an annotated corpus of audio tracks. Each track is represented as a set of feature vectors that is extracted by passing a short-time window over the audio signal. The text description of a track is represented by an *annotation vector*, a vector of weights where each element indicates how strongly a semantic concept (i.e., a word) applies to the audio track.

Our probabilistic model is one *word-level* distributions over the audio feature space for each word in our vocabulary. Each distribution is modeled using a multivariate Gaussian mixture model (GMM). The parameters of a word-level GMM are estimated using audio content from a set of training tracks that are positively associated with the word. Using this model, we can infer likely semantic annotations given a novel track and can use a text-based query to rank-order a set of unannotated tracks. For illustrative purposes, Table Table III.1 displays annotations of songs produced by our system. Placing the most likely words from specific semantic categories into a natural language context demonstrates how our annotation system can be used to generate automatic music reviews. Table Table III.7 shows some of the top songs that the system retrieves from our data set, given various text-based queries.

Our model is based on the supervised multi-class labeling (SML) model that has been recently proposed for the task of image annotation and retrieval by Carneiro and Vasconcelos [19]. They show that their *mixture hierarchies* Expectation Maximization (EM) algorithm [115], used for estimating the parameters of the word-level GMMs, is superior to traditional parameter estimation techniques in terms of computational scalability and annotation performance. We confirm these findings for audio data and extend this estimation technique to handle real-valued

Table III.1 Automatic annotations generated using the audio content. Words in **bold** are output by our system and then placed into a manually-constructed natural language template.

Frank Sinatra - Fly me to the moon
This is a **jazz**y, **singer / songwriter** song that is **calming** and **sad**. It features **acoustic guitar**, **piano**, **saxophone**, a nice **male vocal solo**, and **emotional**, **high-pitched vocals**. It is a song with a **light beat** and a **slow tempo** that you might like listen to while **hanging with friends**.
Creedence Clearwater Revival - Travelin' Band
This is a **rock**in', **classic rock** song that is **arousing** and **powerful**. It features **clean electric guitar**, **backing vocals**, **distorted electric guitar**, a nice **distorted electric guitar solo**, and **strong, duet vocals**. It is a song with a **catchy** feel and is **very danceable** that you might like listen to while **driving**.
New Order - Blue Monday
This is a **pop**py, **electronica** song that is **not emotional** and **not tender**. It features **sequencer**, **drum machine**, **synthesizer**, a nice **male vocal solo**, and **altered with effects**, **high-pitched** vocals. It is a song with a **synthesized texture** and with **positive feelings** that you might like listen to while **at a party**.
Dr. Dre (feat. Snoop Dogg) - Nuthin' but a 'G' thang
This is **dance pop**py, **hip-hop** song that is **arousing** and **exciting**. It features **drum machine**, **backing vocals**, **male vocal**, a nice **acoustic guitar solo**, and **rapping**, **strong** vocals. It is a song that is **very danceable** and with a **heavy beat** that you might like listen to while **at a party**.

(rather than binary) class labels. Real-valued class labels are useful in the context of music since the strength of association between a word and a song is not always all or nothing. For example, based on a study described below, we find that three out of four college students annotate Elvis Presley's "Heartbreak Hotel" as being a 'blues' song while everyone identified B.B. King's "Sweet Little Angel" as being a blues song. Our *weighted* mixture hierarchies EM algorithm explicitly models these respective strengths of associations when estimating the parameters of a GMM.

The semantic annotations used to train our system come from a user

study in which we asked participants to annotate songs using a standard survey. The survey contained questions related to different semantic categories, such as emotional content, genre, instrumentation, and vocal characterizations. The music data used is a set of 500 'Western popular' songs from 500 unique artists, each of which was reviewed by a minimum of three individuals. Based on the results of this study, we construct a vocabulary of 174 'musically-relevant' semantic keywords. The resulting annotated music corpus, referred to as the *Computer Audition Lab 500* (CAL500) data set, is publicly-available[1] and may be used as a common test bed for future research involving semantic music annotation and retrieval.

Though the focus of this work is on music, our system can be used to model other classes of audio data and is scalable in terms of both vocabulary size and training set size. We demonstrate that our system can successfully annotate and retrieve sound effects using a corpus of 1305 tracks and a vocabulary containing 348 words.

The following section discusses how this work fits into the field of music information retrieval (MIR) and relates to research on semantic image annotation and retrieval. Sections III.D and III.E formulate the related problems of semantic audio annotation and retrieval, present the SML model, and describe three parameter estimation techniques including the *weighted* mixture hierarchies algorithm. Section III.F describes the collection of human annotations for the CAL500 data set. Section III.G describes the sound effects data set. Section III.H reports qualitative and quantitative results for annotation and retrieval of music and sound effects. The final section outlines a number of future directions for this research.

III.C Related work

A central goal of the music information retrieval community is to create systems that efficiently store and retrieve songs from large databases of musical content [50]. The most common way to store and retrieve music uses metadata

[1]The CAL-500 data set can be downloaded from http://cosmal.ucsd.edu/cal.

such as the name of the composer or artist, the name of the song or the release date of the album. We consider a more general definition of musical metadata as any non-acoustic representation of a song. This includes genre and instrument labels, song reviews, ratings according to bipolar adjectives (e.g., happy/sad), and purchase sales records. These representations can be used as input to collaborative filtering systems that help users search for music. The drawback of these systems is that they require a novel song to be *manually* annotated before it can be retrieved.

Another retrieval approach, called *query-by-similarity*, takes an audio-based query and measures the similarity between the query and all of the songs in a database [50]. A limitation of query-by-similarity is that it requires a user to have a useful audio exemplar in order to specify a query. For cases in which no such exemplar is available, researchers have developed *query-by-humming* [30], *-beatboxing* [3], and *-tapping* [39]. However, it can be hard, especially for an untrained user, to emulate the tempo, pitch, melody, and timbre well enough to make these systems viable [30]. A natural alternative is to describe music using words, an interface that is familiar to anyone who has used an Internet search engine. A good deal of research has focused on content-based classification of music by genre [76], emotion [68], and instrumentation [41]. These classification systems effectively 'annotate' music with class labels (e.g., 'blues', 'sad', 'guitar'). The assumption of a predefined taxonomy and the explicit labeling of songs into (mutually exclusive) classes can give rise to a number of problems [83] due to the fact that music is inherently subjective.

We propose a content-based *query-by-text* audio retrieval system that learns a relationship between acoustic features and words from a data set of annotated audio tracks. Our goal is to create a more general system that directly models the relationship between audio content and a vocabulary that is less constrained than existing content-based classification systems. The query-by-text paradigm has been largely influenced by work on the similar task of image annotation. We adapt a supervised multi-class labeling (SML) model [19] since it has performed

well on the task of image annotation. This approach views semantic annotation as one multi-class problem rather than a set of binary one-vs-all problems. A comparative summary of alternative supervised one-vs-all [44] and unsupervised ([14, 42]) models for image annotation is presented in [19].

Despite interest within the computer vision community, there has been relatively little work on developing 'query-by-text' for audio (and specifically music) data. One exception is the work of Whitman et al. ([124–126]). Our approach differs from theirs in a number of ways. First, they use a set of web-documents associated with an *artist* whereas we use multiple *song* annotations for each song in our corpus. Second, they take a one-vs-all approach and learn a discriminative classifier (a support vector machine or a regularized least-squares classifier) for each word in the vocabulary. The disadvantage of the one-vs-all approach is that it results in binary decisions for each word. We propose a generative multi-class model that outputs a semantic multinomial distribution over the vocabulary for each song. As we show in Section III.D, the parameters of the multinomial distribution provide a natural ranking of words [19]. In addition, semantic multinomials are a compact representation of an audio track which is useful for efficient retrieval.

Other query-by-text audition systems ([18, 102]) have been developed for annotation and retrieval of sound effects. Slaney's Semantic Audio Retrieval system ([101, 102]) creates separate hierarchical models in the acoustic and text space, and then makes links between the two spaces for either retrieval or annotation. Cano and Koppenberger propose a similar approach based on nearest neighbor classification [18]. The drawback of these non-parametric approaches is that inference requires calculating the similarity between a query and every training example. We propose a parametric approach that requires one model evaluation per semantic concept. In practice, the number of semantic concepts is orders of magnitude smaller than the number of potential training data points, leading to a more scalable solution.

III.D Semantic audio annotation and retrieval

This section formalizes the related tasks of semantic audio annotation and retrieval as a supervised multi-class, multi-label classification problem where each word in a vocabulary represents a class and each song is labeled with multiple words. We learn a *word-level* (i.e., class-conditional) distribution for each word in a vocabulary by training only on the audio tracks that are positively associated with that word. A schematic overview of our model is presented in Figure Figure III.1.

III.D.1 Problem formulation

Consider a vocabulary $\mathcal{V}$ consisting of $|\mathcal{V}|$ unique words. Each 'word' $w_i \in \mathcal{V}$ is a semantic concept such as 'happy', 'blues', 'electric guitar', 'creaky door', etc. The goal in annotation is to find a set $\mathcal{W} = \{w_1, ..., w_A\}$ of A semantically meaningful words that describe a query audio track s_q. Retrieval involves rank ordering a set of tracks (e.g., songs) $\mathcal{S} = \{s_1, ..., s_R\}$ given a set of query words $\mathcal{W}_q$. It will be convenient to represent the text data describing each song as an *annotation* vector $\mathbf{y} = (y_1, ..., y_{|\mathcal{V}|})$ where $y_i > 0$ if w_i has a positive semantic association with the audio track and $y_i = 0$ otherwise. The y_i's are called *semantic weights* since they are proportional to the strength of the semantic association. If the semantic weights are mapped to $\{0, 1\}$, then they can be interpreted as class labels. We represent an audio track s as a set $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$ of T real-valued feature vectors, where each vector $\mathbf{x}_t$ represents features extracted from a short segment of the audio content and T depends on the length of the song. Our data set $\mathcal{D}$ is a collection of track-annotation pairs $\mathcal{D} = \{(\mathcal{X}_1, \mathbf{y}_1), ..., (\mathcal{X}_{|\mathcal{D}|}, \mathbf{y}_{|\mathcal{D}|})\}$.

III.D.2 Annotation

Annotation can be thought of as a multi-class classification problem in which each word $w_i \in \mathcal{V}$ represents a class and the goal is to choose the best class(es) for a given song. Our approach involves modeling a word-level distribution over

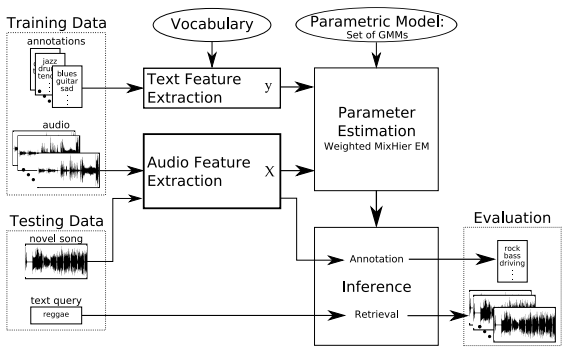


Figure III.1 Semantic annotation and retrieval model diagram.

an audio feature space, $P(\mathbf{x}|i), i \in \{1, ..., |\mathcal{V}|\}$ for each word $w_i \in \mathcal{V}$. Given a track represented by the set of audio feature vectors $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$, we use Bayes' rule to calculate the posterior probability of each word in the vocabulary, given the audio features:

$$P(i|\mathcal{X}) = \frac{P(\mathcal{X}|i)P(i)}{P(\mathcal{X})}, \tag{III.1}$$

where $P(i)$ is the prior probability that word w_i will appear in an annotation. If we assume that $\mathbf{x}_a$ and $\mathbf{x}_b$ are conditionally independent given word w_i (i.e., $\mathbf{x}_a \perp \mathbf{x}_b|w_i, \forall a, b \in \mathcal{N} \leq T, a \neq b$) , then

$$P(i|\mathcal{X}) = \frac{\left[\prod_{t=1}^{T} P(\mathbf{x}_t|i)\right] \cdot P(i)}{P(\mathcal{X})}. \tag{III.2}$$

The naïve Bayes assumption implies that there is no temporal relationship between audio features. While this assumption of conditional independence is unrealistic, attempting to model the temporal interaction between feature vectors may be infeasible due to computational complexity and data sparsity. We assume a uniform prior, $P(i) = 1/|\mathcal{V}|$, for all $i = 1, .., |\mathcal{V}|$ since the T factors in the product will dominate the word prior. We estimate the song prior $P(\mathcal{X})$ by $\sum_{v=1}^{|\mathcal{V}|} P(\mathcal{X}|v)P(v)$ and arrive at our final *annotation* equation:

$$P(i|\mathcal{X}) = \frac{\prod_{t=1}^{T} P(\mathbf{x}_t|i)}{\sum_{v=1}^{|\mathcal{V}|} \prod_{t=1}^{T} P(\mathbf{x}_t|v)}. \tag{III.3}$$

Note that by assuming a uniform word prior, the $1/|\mathcal{V}|$ factor cancels out of the equation.

Using word-level distributions ($P(\mathbf{x}|i)$, $\forall i = 1, ..., |\mathcal{V}|$) and Bayes' rule, we use Equation III.3 to calculate the parameters of a *semantic multinomial* distribution over the vocabulary. That is, each song in our database is compactly represented as a vector of posterior probabilities $\mathbf{p} = \{p_1, ..., p_{|\mathcal{V}|}\}$ in a 'semantic space', where $p_i = P(i|\mathcal{X})$ and $\sum_i p_i = 1$. An example of such a semantic multinomial is given in Figure Figure III.2. To annotate a track with the A best words, we use the word-level models to generate the track's semantic distribution and then choose the A largest peaks of the multinomial distribution, i.e., the A words with maximum posterior probability.

III.D.3 Retrieval

Given the one-word query string w_q, a straightforward approach to retrieval involves ranking songs by $P(\mathcal{X}|q) = \prod_{t=1}^{T} P(\mathbf{x}_t|q)$, where again we use the word-level distribution $P(\mathbf{x}|q)$ and make the naïve Bayes assumption. However, we find empirically that this approach returns almost the same ranking for every word in our vocabulary. The first reason for this is *length bias*: longer tracks (with more feature vectors) have lower likelihoods resulting from the product of additional probability terms (i.e., T is larger). It has been argued that the underestimation of the likelihood is due to the poor naïve Bayes assumption [93]. The second, more subtle, problem is due to the fact that many word-level distributions $P(\mathbf{x}|q)$ are similar (in the Kullback-Leibler sense) to the generic distribution $P(\mathbf{x})$ over the audio feature vector space[2]. This creates a *track bias* in which generic tracks that have high likelihood under this generic distribution will also have high likelihood under many of the word-level distributions.

Both of these problems can be solved by dividing $P(\mathcal{X}|q)$ by the track prior $P(\mathcal{X})$ to normalize for both length and track bias. Note that, if we assume a

[2]This may be caused by using a general purpose audio feature representation that captures additional information besides the specific semantic notion that we are attempting to model. For example, since most of the songs in our training corpus feature vocals, guitar, bass and drums, we would expect most Rolling Stones' songs to be more likely than most Louis Armstrong songs with respect to both the generic distribution $P(\mathbf{x})$ and most word-level distributions $P(\mathbf{x}|q)$.

uniform word prior (which doesn't affect the relative ranking), this is equivalent to ranking by $P(q|\mathcal{X})$ which is calculated in Equation III.3 during annotation. That is, we first annotate our audio corpus by estimating the parameters of a semantic multinomial for each track. Once we have a query multinomial, we rank all the songs in our database by the Kullback-Leibler (KL) divergence between the query multinomial $\mathbf{q}$ and each semantic multinomial. The KL divergence between $\mathbf{q}$ and a semantic multinomial $\mathbf{p}$ is given by [27]:

$$KL(\mathbf{q}||\mathbf{p}) = \sum_{i=1}^{|\mathcal{V}|} q_i \log \frac{q_i}{p_i}, \tag{III.4}$$

where the query distribution serves as the 'true' distribution. Since $q_i = \epsilon$ is effectively zero for all words that do not appear in the query string, a one-word query w_i reduces to ranking by the i-th parameter of the semantic multinomials. For a multiple-word query, we only need to calculate one term in Equation III.4 per word in the query. This leads to a very efficient and scalable approach for music retrieval in which the majority of the computation involves sorting the D scalar KL divergences between the query multinomial and each song in the database.

III.E Parameter Estimation

For each word $w_i \in \mathcal{V}$, we learn the parameters of the word-level (i.e., class-conditional) distribution, $P(\mathbf{x}|i)$, using the audio features from all tracks that have a positive association with word w_i. Each distribution is modeled with a R-component mixture of Gaussians distribution parameterized by $\{\pi_r, \mu_r, \Sigma_r\}$ for $r = 1, ..., R$. The word-level distribution for word w_i is given by:

$$P(\mathbf{x}|i) = \sum_{r=1}^{R} \pi_r \mathcal{N}(\mathbf{x}|\mu_r, \Sigma_r),$$

where $\mathcal{N}(\cdot|\mu, \Sigma)$ is a multivariate Gaussian distribution with mean μ, covariance matrix Σ, and mixing weight π_r. In this work, we consider only diagonal covariance matrices since using full covariance matrices can cause models to overfit the training

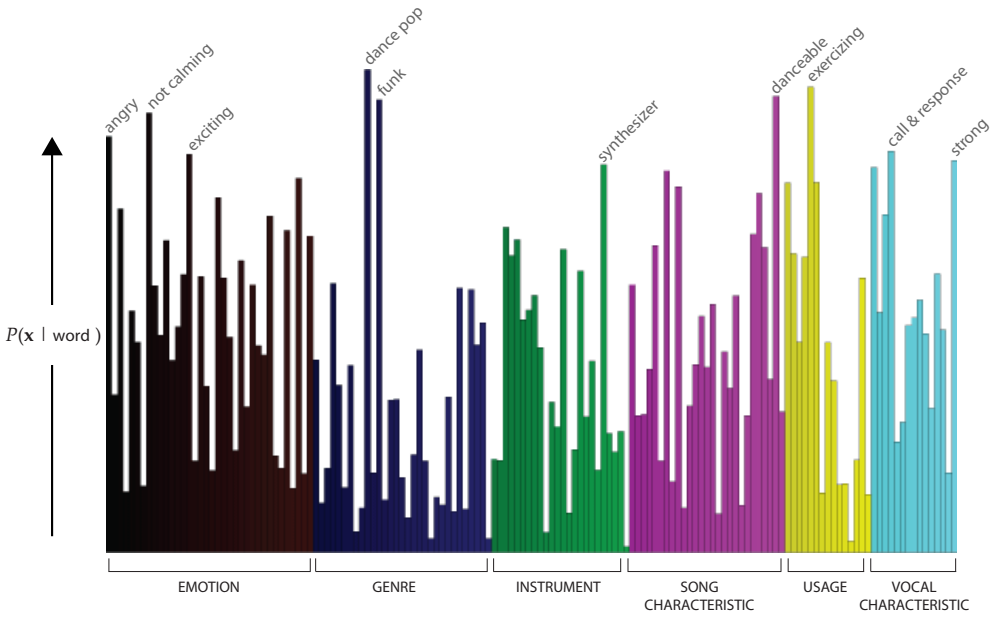


Figure III.2 Semantic multinomial distribution over all words in our vocabulary for the Red Hot Chili Pepper's "Give it Away". Word categories are indicated by color. The 10 most probable words are labeled.

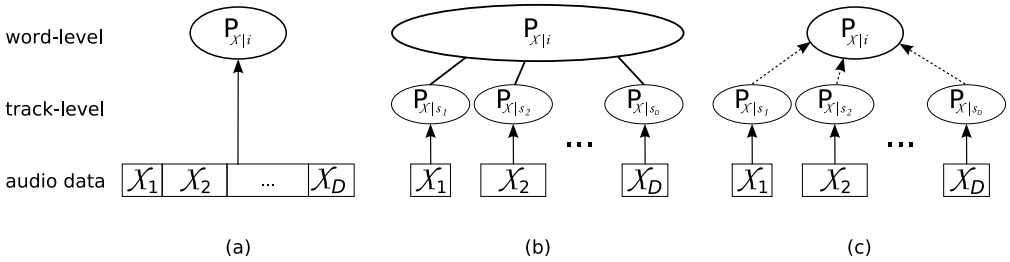


Figure III.3 (a) Direct, (b) naive averaging, and (c) mixture hierarchies parameter estimation. Solid arrows indicate that the distribution parameters are learned using standard EM. Dashed arrows indicate that the distribution is learned using mixture hierarchies EM.

data while scalar covariances do not provide adequate generalization. The resulting set of $|\mathcal{V}|$ models each have $\mathcal{O}(R \cdot D)$ parameters, where D is the dimension of feature vector $\mathbf{x}$.

We consider three parameter estimation techniques for learning the parameters of a word-level distributions: direct estimation, (weighted) modeling averaging, and (weighted) mixture hierarchies estimation. The techniques are similar in that, for each word-level distribution, they use the Expectation-Maximization (EM) algorithm for fitting a mixture of Gaussians to training data. They differ in how they break down the problem of parameter estimation into subproblems and then merge these results to produce a final density estimate.

III.E.1 Direct estimation

Direct estimation trains a model for each word w_i using the superset of feature vectors for all the songs that have word w_i in the associated human annotation: $\bigcup \mathcal{X}_d$, $\forall d$ such that $[\mathbf{y}_d]_i > 0$. Using this training set, we directly learn the word-level mixture of Gaussians distribution using the EM algorithm (see Figure Figure III.3a). The drawback of using this method is that computational complexity increases with training set size. We find that, in practice, we are unable

to estimate parameters using this method in a reasonable amount of time since there are on the order of 100,000's of training vectors for each word-level distribution. One suboptimal work around to this problem is to simply ignore (i.e., subsample) part of the training data.

III.E.2 Model averaging

Instead of directly estimating a word-level distribution for w_i, we can first learn *track-level* distributions, $P(\mathbf{x}|i,d)$ for all tracks d such that $[\mathbf{y}_d]_i > 0$. Here we use EM to train a track-level distribution from the feature vectors extracted from a single track. We then create a word-level distribution by calculating a weighted average of all the track-level distributions where the weights are set by how strongly each word w_i relates to that track:

$$P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|i) = \frac{1}{C}\sum_{d=1}^{|\mathcal{D}|}[\mathbf{y}_d]_i \sum_{k=1}^{K}\pi_k^{(d)}\mathcal{N}(\mathbf{x}|\mu_k^{(d)},\Sigma_k^{(d)}),$$

where $C = \sum_d [\mathbf{y}_d]_i$ is the sum of the semantic weights associated with word w_i, $|\mathcal{D}|$ is total number of training examples, and K is the number of mixture components in each track-level distribution (see Figure Figure III.3b).

Training a model for each track in the training set and averaging them is relatively efficient. The drawback of this non-parametric estimation technique is that the number of mixture components in the word-level distribution grows with the size of the training database since there will be K components for each track-level distribution associated with word w_i. In practice, we may have to evaluate thousands of multivariate Gaussian distributions for each of the feature vectors $\mathbf{x}_t \in \mathcal{X}_q$ of a novel query track, $\mathcal{X}_q$. Note that $\mathcal{X}_q$ may contain thousands of feature vectors depending on the audio representation.

III.E.3 Mixture hierarchies estimation

The benefit of direct estimation is that it produces a distribution with a fixed number of parameters. However, in practice, parameter estimation is infeasible

without subsampling the training data. Model averaging efficiently produces a distribution but it is computationally expensive to evaluate this distribution since the number of parameters increases with the size of the training data set. Mixture hierarchies estimation is an alternative that efficiently produce a word-level distribution with a fixed number of parameters [115].

Consider the set of $|\mathcal{D}|$ track-level distributions (each with K mixture components) that are learned during model averaging estimation for word w_i. We can estimate a word-level distribution with R components by combining the $|\mathcal{D}| \cdot K$ track-level components using the mixture hierarchies EM algorithm. (see Figure Figure III.3c). This EM algorithm iterates between the E-step and the M-step as follows:

E-step: Compute the responsibilities of each word-level component r to a track-level component k from track d:

$$h^r_{(d),k} = \frac{[\mathbf{y}_d]_i \left[\mathcal{N}(\mu^{(d)}_k|\mu_r, \mathbf{\Sigma}_r) e^{-\frac{1}{2}\mathrm{Tr}\{(\mathbf{\Sigma}_r)^{-1}\mathbf{\Sigma}^{(d)}_k\}}\right]^{\pi^{(d)}_k N} \pi_r}{\sum_l \left[\mathcal{N}(\mu^{(d)}_k|\mu_l, \mathbf{\Sigma}_l) e^{-\frac{1}{2}\mathrm{Tr}\{(\mathbf{\Sigma}_l)^{-1}\mathbf{\Sigma}^{(d)}_k\}}\right]^{\pi^{(d)}_k N} \pi_l},$$

where N is a user defined parameter. In practice, we set $N = K$ so that on average $\pi^{(d)}_k N$ is equal to 1.

M-step: Update the parameters of the word-level distribution

$$\begin{aligned}
\pi^{new}_r &= \frac{\sum_{(d),k} h^r_{(d),k}}{W \cdot K}, , \text{ where } W = \sum_{d=1}^{|\mathcal{D}|} [\mathbf{y}_d]_i \\
\mu^{new}_r &= \sum_{(d),k} z^r_{(d),k} \mu^{(d)}_k, \quad \text{where } z^r_{(d),k} = \frac{h^r_{(d),k}\pi^{(d)}_k}{\sum_{(d),k} h^r_{(d),k}\pi^{(d)}_k}, \\
\mathbf{\Sigma}^{new}_r &= \sum_{(d),k} z^r_{(d),k} \left[\mathbf{\Sigma}^{(d)}_k + (\mu^{(d)}_k - \mu_t)(\mu^{(d)}_k - \mu_t)^T\right].
\end{aligned}$$

From a generative perspective, a track-level distribution is generated by sampling *mixture components* from the word-level distribution. The observed audio features are then samples from the track-level distribution. Note that the number of

parameters for the word-level distribution is the same as the number of parameters resulting from direct estimation yet we learn this model using all of the training data without subsampling. We have essentially replaced one computationally expensive (and often impossible) run of the standard EM algorithm with $|\mathcal{D}|$ computationally inexpensive runs and one run of the mixture hierarchies EM. In practice, mixture hierarchies EM requires about the same computation time as one run of standard EM.

Our formulation differs from that derived in [115] in that the responsibility, $h^r_{(d),k}$, is multiplied by the semantic weight $[\mathbf{y}_d]_i$ between word w_i and audio track s_d. This *weighted mixture hierarchies algorithm* reduces to the standard formulation when the semantic weights are either 0 or 1. The semantic weights can be interpreted as a relative measure of importance of each training data point. That is, if one data point has a weight of 2 and all others have a weight of 1, it is as though the first data point actually appeared twice in the training set.

III.F Semantically Labeled Music Data

Perhaps the fastest and most cost effective way to collect semantic information about music is to mine web documents that relate to songs, albums or artists [109, 126]. Whitman et al. collect a large number webpages related to the artist when attempting to annotate individual songs [126]. One drawback of this methodology is that it produces the same training annotation vector for all songs by a single artist. This is a problem for many artists, such as Paul Simon and Madonna, who have produced an acoustically diverse set of songs over the course of their careers. In previous work, we take a more song-specific approach by text-mining song reviews written by expert music critics [109]. The drawback of this technique is that critics do not explicitly make decisions about the relevance of each individual word when writing about songs and/or artists. In both works, it is evident that the semantic labels are a noisy version of an already problematic

'subjective ground truth.'

To address the shortcomings of noisy semantic data mined from text-documents, we decided to collect a 'clean' set of semantic labels by asking human listeners to explicitly label songs with acoustically-relevant words. We considered 135 musically-relevant concepts spanning six semantic categories: 29 instruments were annotated as present in the song or not; 22 vocal characteristics were annotated as relevant to the singer or not; 36 genres, a subset of the Codaich genre list [**?**], were annotated as relevant to the song or not; 18 emotions, found by Skowronek et al. [**?**] to be both important and easy to identify, were rated on a scale from one to three (e.g., "not happy", "neutral", "happy"); 15 song concepts describing the acoustic qualities of the song, artist and recording (e.g., tempo, energy, sound quality); and 15 usage terms from [55] (e.g., "I would listen to this song while *driving, sleeping, etc.*").

The music corpus is a selection of 500 Western popular songs from the last 50 years by 500 different artists. This set was chosen to maximize the acoustic variation of the music while still representing some familiar genres and popular artists. The corpus includes 88 songs from the Magnatunes database [45], one from each artist whose songs are not from the classical genre.

To generate new semantic labels, we paid 66 undergraduate students to annotate our music corpus with the semantic concepts from our vocabulary. Participants were rewarded $10 per hour to listen to and annotate music in a university computer laboratory. The computer-based annotation interface contained a MP3 player and an HTML form. The form consisted of one or more radio boxes and/or check boxes for each of our 135 concepts. The form was not presented during the first 30 seconds of song playback to encourage undistracted listening. Subjects could advance and rewind the music and the song would repeat until they completed the annotation form. Each annotation took about 5 minutes and most participants reported that the listening and annotation experience was enjoyable. We collected at least 3 semantic annotations for each of the 500 songs in our music

corpus and a total of 1708 annotations. This annotated music corpus is referred to as the Computer Audition Lab 500 (CAL500) data set.

III.F.1 Semantic Feature Representation

We expand the set of *concepts* to a set of 237 *words* by mapping all bipolar concepts to two individual words. For example, 'tender' gets mapped to 'tender' and 'not tender' so that we can explicitly learn separate models for tender songs and songs that are not tender. Note that, according to the data that we collected, many songs may be annotated as neither tender nor not tender. Other concepts, such as genres or instruments, are mapped directly to a single word.

For each song, we have a collection of human annotations where each annotation is a vector of numbers expressing the response of a subject to a set of words. For each word, the annotator has supplied a response of +1 or -1 if the annotator believes the song is or is not indicative of the word, or 0 if unsure. We take all the annotations for each song and compact them to a single annotation vector by observing the level of agreement over all annotators. Our final semantic weights $\mathbf{y}$ are

$$[\mathbf{y}]_i = \max\left(0, \left[\frac{\#(\text{Positive Votes}) - \#(\text{Negatives Votes})}{\#(\text{Annotations})}\right]_i\right).$$

For example, for a given song, if four annotators have labeled a concept w_i with +1, +1, 0, -1, then $[\mathbf{y}]_i = 1/4$. The semantic weights are used for parameter estimation.

For evaluation purposes, we also create a binary 'ground truth' annotation vector for each song. To generate this vector, we label a song with a word if a minimum of two people vote for the word and there is a high level of agreement ($[\mathbf{y}]_i = .80$) between all subjects. This assures that each positive label is reliable. Finally, we prune all words that are represented by fewer than five songs. This reduces our set of 237 words to a set of 174 words.

III.F.2 Music Feature Representation

Each song is represented as a *bag-of-feature-vectors*: a set of feature vectors where each vector is calculated by analyzing a short-time segment of the audio signal. In particular, we represent the audio with a time series of *Delta-MFCC* feature vectors [17]. A time series of Mel-frequency cepstral coefficient (MFCC) [89] vectors is extracted by sliding a half-overlapping, short-time window (∼23 msec) over the song's digital audio file. A Delta-MFCC vector is calculated by appending the first and second instantaneous derivatives of each MFCC to the vector of MFCCs. We use the first 13 MFCCs resulting in about 5,200 39-dimensional feature vectors per minute of audio content. The reader should note that the SML model (a set of GMMs) ignores the temporal dependencies between adjacent feature vectors within the time series. We find that randomly sub-sampling the set of delta cepstrum feature vectors so that each song is represented by 10,000 feature vectors reduces the computation time for parameter estimation and inference without sacrificing overall performance.

We have also explored a number of alternative feature representations, many of which have shown good performance on the task of genre classification, artist identification, song similarity, and/or cover song identification [36]. These include auditory filterbank temporal envelope [76], dynamic MFCC [76], MFCC (without derivatives), chroma features [40], and fluctuation patterns [85]. While a detailed comparison is beyond the scope of this paper, one difference between these representations is the amount of the audio content that is summarized by each feature vector. For example, a Delta-MFCC vector is computed from less than 80 msec of audio content, a dynamic MFCC vector summarizes MFCCs extracted over 3/4 of a second, and fluctuation patterns can represent information extracted from 6 seconds of audio content. We found that Delta-MFCC features outperformed the other representations with respect to both annotation and retrieval performance.

III.G Semantically Labeled Sound Effects Data

To confirm the general applicability of the SML model to other classes of audio data, we show that we can also annotate and retrieve sound effects. We use the BBC sound effects library which consists of 1305 sound effects tracks [102]. Each track has been annotated with a short 5-10 word caption. We automatically extract a vocabulary consisting of 348 words by including each word that occurs in 5 or more captions. Each caption for a track is represented as a 348-dimensional binary annotation vector where the i-th value is 1 if word w_i is present in the caption, and 0 otherwise. As with music, the audio content of the sound effect track is represented as a time series of Delta-MFCC vectors, though we use a shorter short-time window ($\sim$11.5 msec) when extracting MFCC vectors. The shorter time window is used in an attempt to better represent important inharmonic noises that are generally present in sound effects.

III.H Model evaluation

In this section, we quantitatively evaluate our SML model for audio annotation and retrieval. See Chapter 2 for a comparision of the model described here to other algorithms for automatic music tagging (e.g., SVM [72], Boosting [38] and Dynamic Texture Mixtures [28]). In this Section, we evaluate our two GMM-based SML models and compare them against three baseline models. The parameters for one SML model, denoted 'MixHier', are estimated using the weighted mixture hierarchies EM algorithm. The second SML model, denoted 'ModelAvg', results from weighted modeling averaging. Our three baseline models include a 'Random' lower bound, an empirical upper bound (denoted 'UpperBnd'), and a third 'Human' model that serves as a reference point for how well an individual human would perform on the annotation task.

The 'Random' model samples words (without replacement) from a multinomial distribution parameterized by the word prior distribution, $P(i)$ for $i = 1...|\mathcal{V}|$,

estimated using the observed word counts of a training set. Intuitively, this prior stochastically generates annotations from a pool of the most frequently used words in the training set. The 'UpperBnd' model uses the ground truth to annotated songs. However, since we require that each model use a fixed number of words to annotate each song, if the ground truth annotation contains too many words, we randomly pick a subset of the words from the annotation. Similarly, if the ground truth annotation contains too few words, we randomly add words to the annotation from the rest of the vocabulary.

Lastly, we will compare an individual's annotation against a 'ground truth' annotation that is found by averaging multiple annotations (i.e., an annotation based on group consensus). Specifically, the 'Human' model is created by randomly holding out a single annotation for a song that has been annotated by 4 or more individuals. This model is evaluated against a 'ground truth' that is obtained combining the remaining annotations for that song. (See Section III.F.1 for the details of our summarization process.) It should be noted that each individual annotation uses on average 36 of the 174 words in our vocabulary. Each ground truth annotation uses on average only 25 words since we require a high-level of agreement between multiple independent annotators for a word to be considered relevant. This reflects the fact that music is inherently subjective in that individuals use different words to describe the same song.

III.H.1 Annotation

Using Equation III.3, we annotate all test set songs with 10 words and all test set sound effect tracks with 6 words. Annotation performance is measured using mean *per-word* precision and recall. Per-word precision is the probability that the model correctly uses the word when annotating a song. Per-word recall is the probability that the model annotates a song that should have been annotated with the word. More formally, for each word w, $|w_H|$ is the number of tracks that have word w in the human-generated 'ground truth' annotation. $|w_A|$ is the number of

tracks that our model automatically annotates with word w. $|w_C|$ is the number of 'correct' words that have been used both in the ground truth annotation and by the model. Per-word recall is $|w_C|/|w_H|$ and per-word precision is $|w_C|/|w_A|$[3]. While trivial models can easily maximize one of these measures (e.g., labeling all songs with a certain word or, instead, none of them), achieving excellent precision and recall simultaneously requires a truly valid model.

Mean per-word recall and precision is the average of these ratios over all the words in our vocabulary. It should be noted that these metrics range between 0.0 and 1.0, but one may be upper-bounded by a value less than 1.0 if either the number of words that appear in a ground truth annotation is greater or lesser than the number of words that are output by our model. For example, if our system outputs 10 words to annotate a test song where the ground truth annotation contains 25 words, mean per-word recall will be upper-bounded by a value less than one. The exact upper bounds for recall and precision depend on the relative frequencies of each word in the vocabulary and can be empirically estimated using the 'UpperBnd' model which is described above.

It may seem more straightforward to use *per-song* precision and recall, rather than the per-word metrics. However, per-song metrics can lead to artificially good results if a system is good at predicting the few common words relevant to a large group of songs (e.g., 'rock") and bad at predicting the many rare words in the vocabulary. Our goal is to find a system that is good at predicting all the words in our vocabulary. In practice, using the 10 best words to annotate each of the 500 songs, our system outputs 166 of the 174 words for at least one song.

Table Table III.2 presents quantitative results for music and Table Table III.3 for sound effects. Table Table III.2 also displays annotation results using only words from each of six semantic categories (emotion, genre, instrumentation, solo,

[3]If the model never annotates a song with word w then per-word precision is undefined. In this case, we estimate per-word precision using the empirical prior probability of the word $P(i)$. Using the prior is similar to using the 'Random' model to estimate the per-word precision, and thus, will in general hurt model performance. This produces a desired effect since we are interested in designing a model that annotates songs using many words from our vocabulary.

usage and vocal). All reported results are means and standard errors computed from 10-fold cross-validation (i.e., 450-song training set, 50-song test set).

The quantitative results demonstrate that the SML models trained using model averaging (ModelAvg) and mixture hierarchies estimation (MixHier) significantly outperform the random baselines for both music and sound effects. For music, MixHier significantly outperforms ModelAvg in both precision and recall when considering the entire vocabulary as well as shows superior performance for most semantic categories, where'instrumentation precision' is the sole exception. However, for sound effects, ModelAvg significantly outperforms MixHier. This might be explained by interpreting model averaging as a non-parametric approach in which the likelihood of the query track is computed under every track-level model in the database. For our sound effects data set, it is often the case that semantically related pairs of tracks are acoustically very similar causing that one track-level model to dominate the average.

Over the entire music vocabulary, the MixHier model performance is comparable to Human model. It is also interesting to note that MixHier model performance is significantly worse than the Human model performance for the more 'objective' semantic categories (e.g., Instrumentation and Genre) but is comparable for more 'subjective' semantic categories (e.g., Usage and Emotion). We are surprised by the low Human model precision, especially for some of these more objective categories, when compared against the UpperBnd model. Taking a closer look at precision for individual words, while there are some words with relatively high precision, such as 'male lead vocals' (0.96) and 'drum set' (0.81), there are many words with low precision. Low precision words arise from a number of causes including test subject inattentiveness (due to boredom or fatigue), non-expert test-subjects (e.g., can't detect a 'trombone' in a horn section), instrument ambiguity (e.g., deciding between 'acoustic guitar' vs. 'clean electric guitar'), and our summarization process. For example, consider the word 'clean electric guitar' and the song "Everything she does is magic" by The Police. Given four

test subjects, two subjects positively associate the song with the word because the overall guitar sound is clean, one is unsure, and one says there is no 'clean electric guitar' presumably because, technically, the guitarist makes use of a delay distortion[4]. Our summarization process would not use the word to label this songs despite the fact that half of the subjects used this word to describe the song. In Section III.I, we will discuss both ways to improve the survey process as well as an alternative data collection technique.

III.H.2 Retrieval

For each one-word query w_q in $\mathcal{V}$, we rank-order a test set of songs. For each ranking, we calculate the average precision (AP) [42] and the area under the receiver operating characteristic curve (AROC). Average precision is found by moving down our ranked list of test songs and averaging the precisions at every point where we correctly identify a new song. An ROC curve is a plot of the true positive rate as a function of the false positive rate as we move down this ranked list of songs. The area under the ROC curve (AROC) is found by integrating the ROC curve and is upper-bounded by 1.0. Random guessing in a retrieval task results in an AROC of 0.5. Comparison to human performance is not possible for retrieval since an individual's annotations do not provide a ranking over all retrievable audio tracks. Mean AP and Mean AROC are found by averaging each metric over all the words in our vocabulary (shown Tables Table III.4 and Table III.6).

As with the annotation results, we see that our SML models significantly outperform the random baseline and that MixHier outperforms ModelAvg for music retrieval. For sound effects retrieval, MixHier and ModelAvg are comparable if we consider Mean AROC, but MixHier shows superior performance if we consider Mean AP.

In addition, we extend the evaluation of music retrieval two- and three-word text-based queries, with results shown in Table Table III.5. Table Table III.7

[4]A delay causes the sound to repeatedly echo as the sound fades away, but does not grossly distort the timbre of electric guitar.

shows the top 5 songs retrieved for a number of text-based queries. In addition to being (mostly) accurate, the reader should note that queries, such as 'Tender' and 'Female Vocals', return songs that span different genres and are composed using different instruments. As more words are added to the query string, note that the songs returned are representative of all the semantic concepts in each of the queries.

III.I Discussion

The qualitative annotation and retrieval results in Tables Table III.1 and Table III.7 indicate that our system can produce sensible semantic annotations for an acoustically diverse set of songs and can retrieve relevant songs given a text-based query. When comparing these results with previous results based on models trained using web-mined data [109], it is clear that using 'clean' data (i.e., the CAL500 data set) results in much more intuitive music reviews and search results.

Our goal in collecting the CAL500 data set was to quickly and cheaply collect a small music corpus with reasonably accurate annotations for the purposes of training our SML model. The human experiments were conducted using (mostly) non-expert college students who spent about five minutes annotating each song using our survey. While we think that the CAL500 data set will be useful for future content-based music annotation and retrieval research, it is not of the same quality as data that might be collected using a highly-controlled psychoacoustics experiment. Future improvements would include spending more time training our test subjects and inserting consistency checks so that we could remove inaccurate annotations from test subjects who show poor performance.

In derivative work, we have examined an extension to our data collection process involving vocabulary selection: if a word in the vocabulary is inconsistently used by human annotators, or the word is not clearly represented by the underlying acoustic representation, the word can be considered as *noisy* and should be removed

from the vocabulary to de-noise the modeling process. We explore these issues in [107], whereby we devise vocabulary pruning techniques based on measurements of human agreement and correlation of words with the underlying audio content.

A further extention is described in Chapter 2 where we collect a much larger annotated data set of music using web-based human computation games. Both Herd It and our other web-based game called "Listen Game" [**?**] allow multiple 'annotators' to label music through realtime competition. We consider this to be a more scalable and cost-effective approach for collecting high-quality music annotations than laborious surveys. We are also able to grow our vocabulary by allowing users to suggest words that describe the music.

The weighted mixture hierarchies EM algorithm presented here is more computationally efficient and produces better density estimates than direct estimation or modeling averaging. The improvement in performance may be attributed to the fact that we represent each track with a track-level distribution before modeling a word-level distribution. The track-level distribution is a smoothed representation of the bag-of-feature-vectors that are extracted from the audio signal. We then learn a mixture from the mixture components of the track-level distributions that are semantically associated with a word. The benefit of using smoothed estimates of the tracks is that the EM framework, which is prone to find poor local maxima, is more likely to converge to a better density estimate.

The *semantic multinomial* representation of a song, which is generated during annotation (see Section III.D.2), is a useful and compact representation of a song. We show that if we construct a *query multinomal* based on a multi-word query string, we can quickly retrieve relevant songs based on the Kullback-Liebler (KL) divergence between the query multinomial and all semantic multinomials in our database of automatically annotated tracks. The semantic multinomial representation is also useful for related audio information tasks such as 'retrieval-by-semantic-similarity' [7, 12].

It should be noted that we use a very basic frame-based audio feature

representation. We can imagine using alternative representations, such as those that attempt to model higher-level notions of harmony, rhythm, melody, and timbre. Similarly, our probabilistic SML model (a set of GMMs) is one of many models that have been developed for image annotation [14, 42]. Future work may involve adapting other models for the task of audio annotation and retrieval. In addition, one drawback of our current model is that, by using GMMs, we ignore all temporal dependencies between audio feature vectors. Future research will involve exploring models, such as hidden Markov models, that explicitly model the longer-term temporal aspects of music.

Lastly, an interesting direction for future work will involve modeling individual users (or subsets of similar users) with *user-specific* models. For example, during data collection, we had one test subject annotate 200 of the 500 songs in our data set. A preliminary study showed that we were better able to predict some words (especially 'usage' words) for this subject using the 200-song subset when compared against models trained using the entire CAL500 data set. This is not surprising since we would expect an individual to be *self-consistent* when annotating songs with subjective concepts. We expect that user-specific models, made possible given data such as collected by Herd It (where a user's Facebook profile is associated with each tag) will offer the potential to reduce the impact of subjectivity in music such that we can better model an individual's notions of audio semantics.

III.J Acknowledgements

Chapter 3, in full, is a reprint of the material as it appears in IEEE Transactions on Audio, Speech and Language Processing 16(2). D. Turnbull, L. Barrington and G.R.G Lanckriet, 2008. The dissertation author was investigator and co-author of this paper.

Table III.2 Music annotation results. Track-level models have $K = 8$ mixture components, word-level models have $R = 16$ mixture components. A = annotation length (determined by the user), $|\mathcal{V}|$ = vocabulary size.

Category	A / $\lvert\mathcal{V}\rvert$	Model	Precision		Recall	
All Words	10 / 174	Random	0.144	(0.004)	0.064	(0.002)
		Human	0.296	(0.008)	0.145	(0.003)
		UpperBnd	*0.712*	(0.007)	*0.375*	(0.006)
		ModelAvg	0.189	(0.007)	0.108	(0.009)
		MixHier	**0.265**	(0.007)	**0.158**	(0.006)
Emotion	4 / 36	Random	0.276	(0.012)	0.113	(0.004)
		Human	0.453	(0.014)	0.180	(0.006)
		UpperBnd	*0.957*	(0.005)	*0.396*	(0.010)
		ModelAvg	0.366	(0.012)	0.179	(0.005)
		MixHier	**0.424**	(0.008)	**0.195**	(0.004)
Genre	2 / 31	Random	0.055	(0.005)	0.079	(0.008)
		Human	0.268	(0.017)	0.290	(0.021)
		UpperBnd	*0.562*	(0.026)	*0.777*	(0.018)
		ModelAvg	0.122	(0.012)	0.161	(0.017)
		MixHier	**0.171**	(0.009)	**0.242**	(0.019)
Instrumentation	4 / 24	Random	0.141	(0.009)	0.195	(0.014)
		Human	0.416	(0.014)	0.522	(0.008)
		UpperBnd	*0.601*	(0.015)	*0.868*	(0.018)
		ModelAvg	**0.267**	(0.008)	0.320	(0.022)
		MixHier	0.259	(0.010)	**0.381**	(0.021)
Solo	1/ 9	Random	0.031	(0.007)	0.155	(0.035)
		Human	0.104	(0.020)	0.158	(0.034)
		UpperBnd	*0.197*	(0.019)	*0.760*	(0.052)
		ModelAvg	0.057	(0.012)	0.231	(0.033)
		MixHier	**0.060**	(0.012)	**0.261**	(0.050)
Usage	2 / 15	Random	0.073	(0.008)	0.154	(0.016)
		Human	0.125	(0.012)	0.175	(0.023)
		UpperBnd	*0.363*	(0.014)	*0.814*	(0.031)
		ModelAvg	0.103	(0.010)	0.170	(0.017)
		MixHier	**0.122**	(0.012)	**0.264**	(0.027)
Vocal	2 / 16	Random	0.062	(0.007)	0.153	(0.018)
		Human	0.188	(0.021)	0.304	(0.023)
		UpperBnd	*0.321*	(0.017)	*0.788*	(0.019)
		ModelAvg	0.102	(0.008)	0.226	(0.016)
		MixHier	**0.134**	(0.005)	**0.335**	(0.021)

Table III.3 Sound effects annotation results. $A = 6, |\mathcal{V}| = 348$.

Model	Recall		Precision	
Random	0.018	(0.002)	0.012	(0.001)
UpperBnd	*0.973*	(0.004)	*0.447*	(0.009)
ModelAvg ($K = 4$)	**0.360**	(0.014)	**0.179**	(0.010)
MixHier ($K = 8, R = 16$)	0.306	(0.010)	0.145	(0.005)

Table III.4 Music retrieval results. $|\mathcal{V}| = 174$.

Category	$\|\mathcal{V}\|$	Model	MeanAP		MeanAROC	
All Words	174	Random	0.231	(0.004)	0.503	(0.004)
		ModelAvg	0.372	(0.008)	0.682	(0.006)
		MixHier	**0.390**	(0.004)	**0.710**	(0.004)
Emotion	36	Random	0.327	(0.006)	0.504	(0.003)
		ModelAvg	0.486	(0.013)	0.685	(0.010)
		MixHier	**0.506**	(0.008)	**0.710**	(0.005)
Genre	31	Random	0.132	(0.005)	0.500	(0.005)
		ModelAvg	0.309	(0.020)	0.695	(0.008)
		MixHier	**0.329**	(0.012)	**0.719**	(0.005)
Instrumentation	24	Random	0.221	(0.007)	0.502	(0.004)
		ModelAvg	0.372	(0.015)	0.694	(0.008)
		MixHier	**0.399**	(0.018)	**0.719**	(0.006)
Solo	9	Random	0.106	(0.014)	0.502	(0.004)
		ModelAvg	**0.190**	(0.028)	0.688	(0.008)
		MixHier	0.180	(0.025)	**0.712**	(0.006)
Usage	15	Random	0.169	(0.012)	0.501	(0.005)
		ModelAvg	0.231	(0.012)	0.684	(0.007)
		MixHier	**0.240**	(0.016)	**0.707**	(0.004)
Vocal	16	Random	0.137	(0.006)	0.502	(0.004)
		ModelAvg	0.234	(0.019)	0.680	(0.007)
		MixHier	**0.260**	(0.018)	**0.705**	(0.005)

Table III.5 Music retrieval results for 2- and 3-word queries.

Query Length	Model	MeanAP	MeanAROC
2-words	Random	0.076	0.500
(4,658/15,225)	SML	**0.164**	**0.723**
3-words	Random	0.051	0.500
(50,471/1,756,124)	SML	**0.120**	**0.730**

Table III.6 Sound effects retrieval results. $|\mathcal{V}| = 348$.

Model	Mean AP		Mean AROC	
Random	0.051	(0.002)	0.506	(0.004)
ModelAvg ($K = 4$)	0.183	(0.003)	**0.785**	(0.005)
MixHier ($K = 8, R = 16$)	**0.331**	(0.008)	0.784	(0.006)

Table III.7 Qualitative music retrieval results for our SML model. Results are shown for 1-, 2- and 3-word queries.

Query	Returned Songs
Pop	The Ronettes- Walking in the Rain
	The Go-Gos - Vacation
	Spice Girls - Stop
	Sylvester - You make me feel mighty real
	Boo Radleys - Wake Up Boo!
Female Lead Vocals	Alicia Keys - Fallin'
	Shakira - The One
	Christina Aguilera - Genie in a Bottle
	Junior Murvin - Police and Thieves
	Britney Spears - I'm a Slave 4 U
Tender	Crosby Stills and Nash - Guinnevere
	Jewel - Enter from the East
	Art Tatum - Willow Weep for Me
	John Lennon - Imagine
	Tom Waits - Time
Pop AND Female Lead Vocals	Britney Spears - I'm a Slave 4 U
	Buggles - Video Killed the Radio Star
	Christina Aguilera - Genie in a Bottle
	The Ronettes - Walking in the Rain
	Alicia Keys - Fallin'
Pop AND Tender	5th Dimension - One Less Bell to Answer
	Coldplay - Clocks
	Cat Power - He War
	Chantal Kreviazuk - Surrounded
	Alicia Keys - Fallin'
Female Lead Vocals AND Tender	Jewel - Enter from the East
	Evanescence - My Immortal
	Cowboy Junkies - Postcard Blues
	Everly Brothers - Take a Message to Mary
	Sheryl Crow - I Shall Believe
Pop AND Female Lead Vocals AND Tender	Shakira - The One
	Alicia Keys - Fallin'
	Evanescence - My Immortal
	Chantal Kreviazuk - Surrounded
	Dionne Warwick - Walk on by

Chapter IV

Modeling Music as a Dynamic Texture

IV.A Abstract

We consider representing a short temporal fragment of musical audio as a *dynamic texture*, a model of both the timbral and rhythmical qualities of sound, two of the important aspects required for automatic music analysis. The dynamic texture model treats a *sequence* of audio feature vectors as a sample from a linear dynamical system. We apply this new representation to the task of automatic song segmentation. In particular, we cluster audio fragments, extracted from a song, as samples from a dynamic texture mixture (DTM) model. We show that the DTM model can both accurately cluster coherent segments in music and detect transition boundaries. Moreover, the generative character of the proposed model of music makes it amenable for a wide range of applications besides segmentation. As an example, we use DTM models of songs to suggest possible improvements in some other music information retrieval applications such as music annotation and similarity.

IV.B Introduction

Models of music begin with a representation of the audio content in some machine-readable form. It is common practice in music information retrieval to represent a song as an unordered set or "bag" of audio feature vectors (e.g., Mel-frequency cepstral coefficients). While this has shown promise in many applications, (e.g., music annotation and retrieval [111], audio similarity [7] and song segmentation [5]), the bag-of-feature-vectors representation is fundamentally limited by ignoring the time-dependency between feature vectors (permuting the feature vectors in the bag will not alter the representation, so information encapsulated in how feature vectors are ordered in time is ignored). As a result, the bag-of-feature-vectors representation fails to represent the higher-level, longer-term musical dynamics of an audio fragment, like rhythmic qualities (e.g., tempo and beat patterns) and temporal structure (e.g., repeated riffs and arpeggios).

In this paper, we address the limitations of the bag-of-features representation, by modeling simultaneously the instantaneous spectral content (timbre) as well as the longer-term spectral dynamics (rhythmic and temporal structure) of audio fragments that are several seconds in length [6]. To do this, we propose to use a *dynamic texture* (DT) [34] to represent a *sequence* of audio feature vectors as a sample from a generative probabilistic model, specifically, a linear dynamical system (LDS).

One application where it is useful to model the temporal, as well as timbral, dynamics of music is automatic song segmentation; the task of dividing a song into self-coherent units which a human listener would label as similar (e.g., verse, chorus, bridge, etc.). In particular, we propose a new algorithm that segments a song by clustering fragments of the song's audio content, using a *dynamic texture mixture* (DTM) model [22]. We test the segmentation algorithm on a wide variety of songs from two popular music datasets, and show that the dynamic texture captures much of the information required to determine the structure of music.

We also illustrate the applicability of the DTM segmentation to other music information retrieval problems. For example, one common problem with semantic song annotation (auto-tagging) occurs when different segments of the same song contain a variety of musical styles and instrumentations (the "Bohemian Rhapsody problem"). For such songs, the bag-of-features representation averages musical information from the whole song and existing auto-tagging systems (e.g., [111]) will produce generic descriptions of the song. One solution to this problem is first to segment the song into its constituent parts using the proposed automatic segmentation algorithm, and then to generate tags for each segment. We show that the dynamic texture model produces musical segments with homogeneous timbre and tempo, resulting in a more precise description of the song.

The remainder of this paper is organized as follows. In Section IV.C, we review related work on song segmentation. In Section IV.D, we introduce the dynamic texture models for audio fragments, and in Section IV.E we propose an

algorithm for segmenting song structure using the DTM. Section IV.F evaluates the segmentation algorithm on two music datasets. Finally, Section IV.G illustrates several applications of song segmentation to music annotation, retrieval, and visualization.

IV.C Related Work

The goal of automatic song segmentation is to divide a song into self-coherent units such as the chorus, verse, bridge, etc. Foote [43] segments music based on self-similarity between timbre features. Paulus and Klapuri [87] efficiently search the space of all possible segmentations and use a musicological model to label the most plausible segmentation.

Other methods attempt to model music explicitly and then cast segmentation as a clustering problem. Gaussian mixture models (GMMs) ignore temporal relations between features but model music well for applications such as music segmentation and similarity [5] as well as classification of a variety of semantic musical attributes [111]. Hidden Markov models (HMMs) consider transitions between feature states and have offered improvements for segmentation [66], key phrase detection [70] and genre classification [91]. Abdallah et al. [2] incorporate prior knowledge about segment duration into a HMM clustering model to address the problem of over-segmentation. Levy and Sandler [65] realize that feature-level HMMs do not capture sufficient temporal information so encode musical segments as clusters of HMM state-sequences and improve their clustering using constraints based on the temporal length of musical segments.

The DT model used in this paper is similar to the HMM, in that they are both probabilistic time-series models with hidden states that evolve over time. The main difference between the two models is that the hidden states of the HMM take on *discrete values*, whereas those of the DT are *real-valued vectors*. As a consequence, the HMM representation discretizes the observations into bins defined

by the observation likelihoods, and the evolution of the sequence is modeled as jumps between these bins. The *continuous* state space of the DT, on the other hand, can capture smooth (rather than discrete) dynamics of state transitions and model the observed audio fragments without quantization.

Structural segmentation of music is often used as a first step in discovering distinctive or repeated sections that can serve as a representative summary or musical thumbnail of both acoustic [20, 70, 88] and symbolic [54] music representations. For example, Bartsch and Wakefield [10] follow [43] but use chroma features to identify repeated segments for audio thumbnailing and Goto adds high-level assumptions about repeated sections to build a system for automatically detecting choruses [49].

Similar to song segmentation is the task of detecting boundaries between musical segments (e.g., the change from verse to chorus). Turnbull et al. [112] present both an unsupervised (picking peaks of difference features) and supervised (boosted decision stumps) method for identifying musical segment boundaries. Similarly, Ong and Herrera [81] look for novelty in successive feature vectors to predict segment boundaries. These methods only detect the segment boundaries and make no attempt to assess the similarity of resulting segments.

Our formulation of treating audio as a dynamic texture was originally introduced in [6]. The current paper goes beyond [6] in the following ways: 1) we include a complete description of our segmentation algorithm; 2) we add a new step to the algorithm that uses music-based constraints to smooth the segments; 3) we present additional experiments on the PopMusic dataset from [65], along with illustrative examples; and 4) we include additional and more rigorous experiments on automatic annotation of music segments as opposed to entire songs.

IV.D Dynamic Textures Models

Consider representing the audio fragment in Figure Figure IV.1(a) with the corresponding sequence of audio feature vectors shown in Figure Figure IV.1(b). We would like to use these features to model simultaneously the instantaneous audio content (e.g., the instrumentation and timbre) and the melodic and rhythmic content (e.g., guitar riff, drum patterns, and tempo). In this work, we will model the temporal dependencies in the audio *fragment* using a single model for the entire *sequence* of feature vectors. In particular, we will treat the sequence of feature vectors as a sample from a linear dynamical system (LDS). The LDS contains two random variables: 1) an observed variable, which generates the feature vector at each time-step (i.e., the instantaneous audio); and 2) a hidden variable which models the higher-level musical state and how it dynamically evolves over time (i.e., the melodic and rhythmic content). In this way, we are able to capture *both the spectral and temporal properties* of the musical signal in a single probabilistic generative model.

The treatment of a time-series as a sample from a linear dynamical system is also known as a *dynamic texture* (DT) [34] in the computer vision literature, where a video is modeled as a sequence of vectorized image frames. The dynamic texture model has been successfully applied to various computer vision problems, including video texture synthesis [34], video recognition [21, 97], and motion segmentation [22, 35]. Although the DT was originally proposed in the computer vision literature as a generative model of video sequences, it is a generic model that can be applied to any time-series data, which in our case, are sequences of feature vectors that represent fragments of musical audio.

IV.D.1 Dynamic textures

A dynamic texture [34] is a generative model that treats a vector time-series as a sample from a linear dynamical system (LDS). Formally, the model

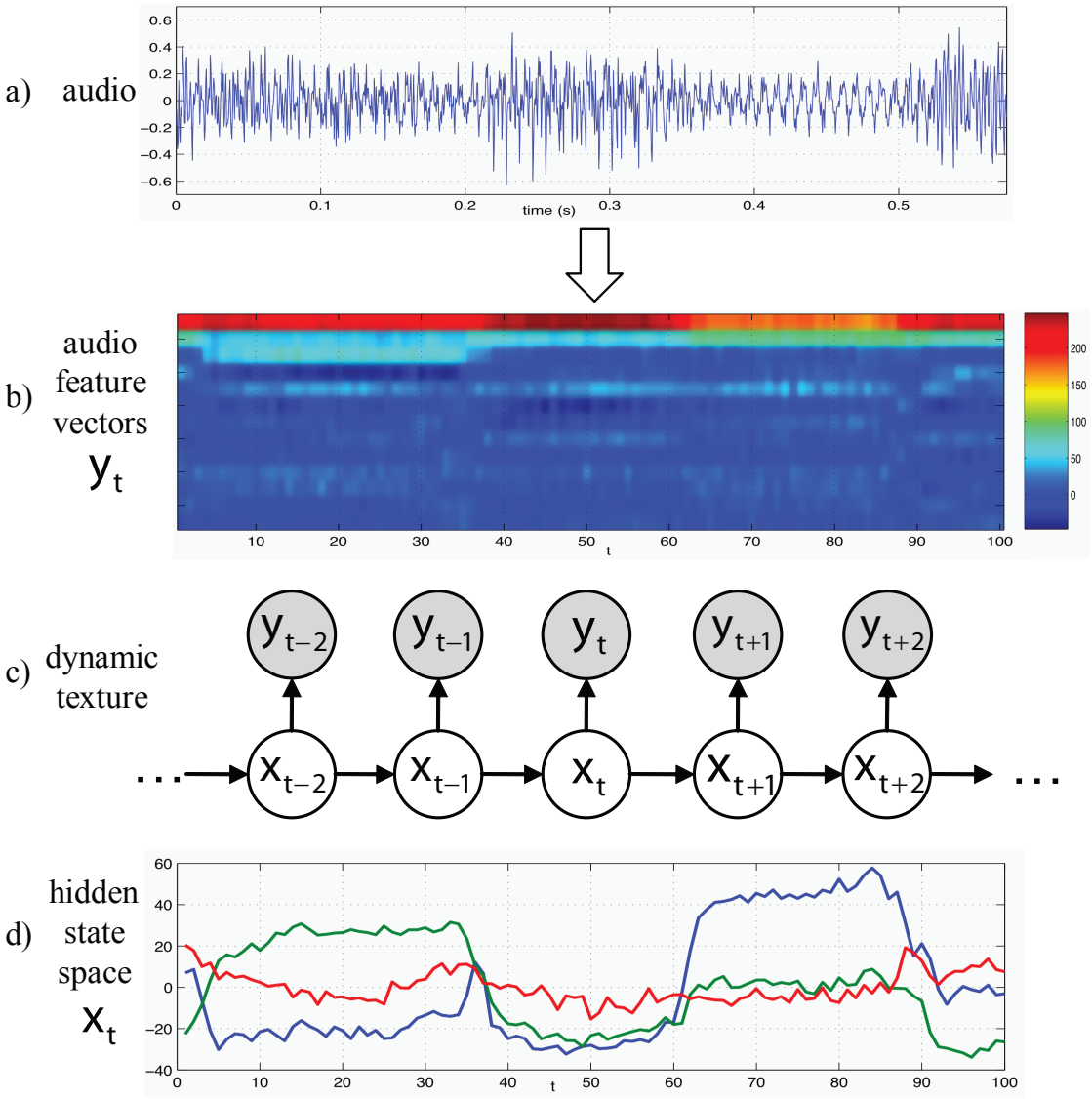


Figure IV.1 Modeling audio as a temporal texture: (a) an audio waveform, and (b) feature vectors y_t extracted from the audio; (c) the sequence of features vectors $\{y_t\}$ is modeled as the output of a linear dynamical system, where (d) the hidden state-space sequence $\{x_t\}$ encodes both the instantaneous sound texture and the evolution of this texture over time.

captures both the appearance and the dynamics of the sequence with two random variables: an *observed variable* $y_t \in \mathbb{R}^m$, which encodes the appearance component (feature vector at time t); and a *hidden state variable* $x_t \in \mathbb{R}^n$ (with $n < m$), which

encodes higher-level characteristics of the time-series and their dynamics (sequence evolution over time). The state and observed variables are related through the *linear dynamical system* (LDS) defined by

$$\begin{cases} x_t = Ax_{t-1} + v_t \\ y_t = Cx_t + w_t + \bar{y} \end{cases}, \tag{IV.1}$$

where $A \in \mathbb{R}^{n \times n}$ is a *state transition matrix*, which encodes the dynamics of the hidden state, $C \in \mathbb{R}^{m \times n}$ is an *observation matrix*, which maps the hidden state variable to an observed feature vector, and $\bar{y} \in \mathbb{R}^m$ is the mean of the observed feature vectors, or the constant offset of the observation variable, y_t. The *driving noise process* v_t is normally distributed with zero mean and covariance Q, i.e., $v_t \sim \mathcal{N}(0, Q)$ where $Q \in \mathbb{S}_+^n$ is a positive definite $n \times n$ matrix, with $\mathbb{S}_+^n$ the set of positive definite matrices of dimension $n \times n$. The *observation noise* w_t is also zero mean and Gaussian, with covariance R, i.e., $w_t \sim \mathcal{N}(0, R)$ where $R \in \mathbb{S}_+^m$. The initial state vector x_1, which determines the starting point of the model, is distributed according to $x_1 \sim \mathcal{N}(\mu, S)$, with $\mu \in \mathbb{R}^n$ and $S \in \mathbb{S}_+^n$. The dynamic texture is specified by parameters $\Theta = \{A, Q, C, R, \mu, S, \bar{y}\}$ and the graphical model of the dynamic texture is shown in Figure Figure IV.1(c).

A number of methods are available to learn the parameters of the dynamic texture from a training sequence, including maximum-likelihood methods (e.g., expectation-maximization [100]), non-iterative subspace methods (e.g., N4SID [82], CCA [11,62]) or a suboptimal, but computationally efficient, least-squares procedure [34]. The dynamic texture has an interesting interpretation when the columns of C are orthogonal (e.g., when learned with the method of [34]). In this case, the columns of C are the principal components of the observations (feature vectors) in time. Hence, the hidden state vector x_t contains the PCA coefficients that generate each observation y_t, where the PCA coefficients (x_t) themselves evolve over time according to a Gauss-Markov process. In this sense, the dynamic texture is an evolving PCA representation of the sequence.

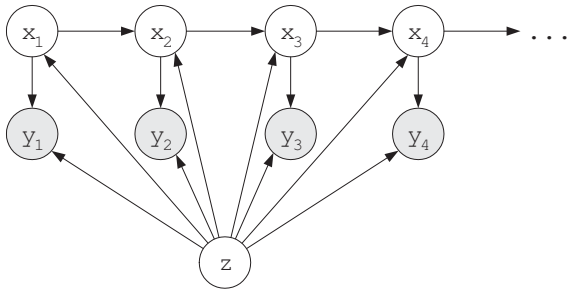


Figure IV.2 Graphical model for the dynamic texture mixture. The hidden variable z selects the parameters of the DT represented by the remaining nodes.

IV.D.2 Mixture of Dynamic Textures

The DT models a single observed sequence, e.g., an audio fragment lasting several seconds. It could also model multiple sequences, if all exhibited the same dynamic texture (specified by the parameters Θ). However, many applications require the simultaneous analysis of N sequences, where it is known a priori that any single sequence exhibits one of a small set of K dynamic textures (with $K << N$). For example, the sequences could be audio fragments extracted from a song that can be clustered into a limited number of textures (e.g., corresponding to the verse, chorus, bridge, etc.). Such a clustering would unravel the verse-chorus-bridge structure of the song. An extension of the DT, the *dynamic texture mixture* (DTM) model, was proposed in [22] to handle exactly this situation. The DTM is a generative model that treats a *collection* of N sequences as samples from a set of K dynamic textures.

Clustering is performed by first learning a DTM for the sequences, and then assigning each sequence to the DT component with largest posterior probability. This is analogous to clustering feature vectors using a Gaussian mixture model (GMM), except that the DTM clusters time-series (sequences of feature vectors), whereas the GMM clusters only feature vectors.

Formally, the DTM [22] is a mixture model where each mixture component

is a dynamic texture, and is defined by the system of equations

$$\begin{cases} x_t = A_z x_{t-1} + v_t \\ y_t = C_z x_t + w_t + \bar{y}_z \end{cases}, \tag{IV.2}$$

where

$$z \sim \text{multinomial}(\alpha_1, \cdots, \alpha_K), \quad \text{s.t.} \sum_{j=1}^{K} \alpha_j = 1 \tag{IV.3}$$

is a random variable that signals the mixture component from which each sequence is drawn. Conditioned on this assignment variable z, the hidden-state x_t and observation y_t behave like a standard dynamic texture with parameters $\Theta_z = \{A_z, Q_z, C_z, R_z, \mu_z, S_z, \bar{y}_z\}$. The graphical model for the dynamic texture mixture is presented in Figure Figure IV.2.

In computer vision, the model has been shown to be a robust model for motion segmentation by clustering patches of video [22]. In this paper, we will use the DTM to segment a song into sections (e.g., verse, chorus, and bridge) in a similar way by clustering audio fragments (sequences of audio feature vectors) extracted from the song. We next present an algorithm for learning the parameters of a DTM from training sequences.

IV.D.3 Parameter estimation of DTMs

Given a set of N sequences $\{y^{(i)}\}_{i=1}^{N}$, where $y^{(i)} = \{y_1^{(i)}, \cdots y_\tau^{(i)}\}$ and τ is the sequence length, the parameters Θ that best fit the observed sequences, in the maximum-likelihood sense [58], can be learned by optimizing

$$\Theta^* = \underset{\Theta}{\text{argmax}} \sum_{i=1}^{N} \log p(y^{(i)}; \Theta), \tag{IV.4}$$

where $\Theta = \{\Theta_j, \alpha_j\}_{j=1}^{K}$ are the parameters of the DTM, and $\Theta_j = \{A_j, Q_j, C_j, R_j, \mu_j, S_j, \bar{y}_j\}$ are the parameters for the j^{th} DT component. Note that the data likelihood function $p(y^{(i)}; \Theta)$ depends on two sets of hidden variables: 1) the assignment variable $z^{(i)}$, which assigns each sequence $y^{(i)}$

to a mixture component; and 2) the hidden state sequence $x^{(i)} = \{x_1^{(i)}, \cdots x_\tau^{(i)}\}$ that produces each $y^{(i)}$. Since the data likelihood depends on hidden variables (i.e., missing information), the maximum-likelihood solution of (IV.4) can be found with recourse to the Expectation Maximization (EM) algorithm [32]. The EM algorithm is an iterative procedure that alternates between estimating the missing information with the current parameters, and computing new parameters given the estimate of the missing information. For the DTM, each iteration of EM consists of

$$\mathrm{E-Step}: \ \mathcal{Q}(\Theta; \hat{\Theta}) = \mathrm{E}_{X,Z|Y;\hat{\Theta}}(\log p(X, Y, Z; \Theta)) \tag{IV.5}$$

$$\mathrm{M-Step}: \ \hat{\Theta}^* = \underset{\Theta}{\operatorname{argmax}}\, \mathcal{Q}(\Theta; \hat{\Theta}) \tag{IV.6}$$

where $p(X, Y, Z; \Theta)$ is the complete-data likelihood of the observations $Y = \{y^{(i)}\}_{i=1}^N$, hidden states sequences $X = \{x^{(i)}\}_{i=1}^N$, and hidden assignment variables $Z = \{z^{(i)}\}_{i=1}^N$, parameterized by Θ.

The EM algorithm for the mixture of dynamic textures was derived in [22], and a summary is presented in Algorithm 1. The E-step relies on the Kalman smoothing filter [22, 100] to compute: 1) the expectations of the hidden state variables x_t, given the observed sequence $y^{(i)}$ came from the j^{th} component; and 2) the likelihood of observing $y^{(i)}$ from the j^{th} component. The M-step then computes the maximum-likelihood parameter values for each dynamic texture component j, by averaging over all sequences $\{y^{(i)}\}_{i=1}^N$, weighted by the posterior probability of assigning $z^{(i)} = j$.

It is known that the accuracy of parameter estimates produced by EM is dependent on how the algorithm is initialized. We use the initialization strategy from [22], where EM is run several times with an increasing number of mixture components. After each EM converges, one of the components is duplicated and its parameters are perturbed slightly, and EM is run again on the new mixture model. More details on the EM algorithm for DTM and the initialization strategy are available in [22].

IV.E Song Segmentation with DTM

Figure Figure IV.3 outlines our approach to song segmentation using the DTM model. First, audio features vectors are extracted from the song's audio waveform (e.g., Mel-frequency cepstral coefficients shown in Figure Figure IV.3(b)). Overlapping sequences of audio feature vectors are extracted from a 5 second fragment of the song where the start position of the fragment slides through the entire song with a large step-size (∼0.5sec). A DTM is learned from the collection of these audio fragments and a **coarse** song segmentation is obtained by assigning each 5 second audio fragment to the most probable DTM component (Figure **??**(c)). Next, we **constrain** the assigned segmentation so that very short segments are unlikely (Figure Figure IV.3(d)). Finally, we run a second segmentation using sequences with a much smaller fragment length (∼1.75sec) and step size (∼0.05sec) to **refine** the precise location of the segment boundaries (Figure Figure IV.3(e)) and evaluate the results with reference to a human-labeled "true" segmentation (Figure Figure IV.3(f)). Each of these steps is described in detail below.

IV.E.1 Features

The content of each 22,050Hz-sampled, monaural waveform is represented using two types of music information features:

Mel-Frequency Cepstral Coefficients

Mel-frequency cepstral coefficients (MFCCs), developed for speech analysis [89], describe the timbre or spectral shape of a short-time piece of audio and are a popular feature for a number of music information analysis tasks, including segmentation [5, 43, 112]. We compute the first 13 MFCCs for half-overlapping frames of 256 samples (each feature vector summarizes 12msec of audio, extracted every 6msec). In music information retrieval, it is common to augment the MFCC feature vector with its instantaneous first and second derivatives, in order to capture

some information about the temporal evolution of the feature. When using the DT, this extra complexity is not required since the temporal evolution is modeled explicitly by the DT.

Chroma

Chroma features have also been successfully applied for song segmentation [10, 49, 80]. They represent the harmonic content of a short-time window of audio by computing the spectral energy present at frequencies that correspond to each of the 12 notes and their octave harmonics in a standard chromatic scale. We compute a 12-dimensional chroma feature vector from three-quarter overlapping frames of 2048 samples (each feature vector summarizes 93msec of audio, extracted every 23msec).

IV.E.2 Song segmentation

Song segmentation is performed with the DTM using a coarse-to-fine approach. A DTM is learned from the collection of audio fragments, using the EM algorithm described in Section IV.D.3. A coarse song segmentation is formed by assigning each fragment to the DTM component with largest posterior probability, i.e.,

$$j^{*(i)} = \underset{j}{\operatorname{argmax}} \frac{\alpha_j p(y^{(i)}; \Theta_j)}{\sum_{j=1}^{K} \alpha_j p(y^{(i)}; \Theta_j)} \tag{IV.7}$$

where $p(y^{(i)}; \Theta_j)$ is the likelihood of sequence $y^{(i)}$ under the j-th mixture component Θ_j. Next, musical constraints are applied to the segmentation, and the boundaries are refined for better localization.

IV.E.3 Musical Constraints on Segments

Levy and Sandler [65] note that musical segments are most likely to last 16 or 32 beats (4 or 8 bars of music in standard 4/4 time). They find that imposing constraints on the minimum segment length results in improved segmentations.

To include this constrained clustering in our model, we wish to encourage audio fragments which are close in time to be assigned to the same segment class. This defines a Markov random field (MRF) over the DTM's assignment variables, Z, which restricts the probability that $z^{(i)}$, the class label variable for a given output $y^{(i)} \in Y$, will differ from the labels assigned to sequences neighboring $y^{(i)}$.

The MRF penalizes the class conditional likelihoods output by the DTM in proportion to their disagreement with the class labels assigned to neighboring sequences. The constrained assignments are estimated as in [65] using iterated conditional modes (ICM) as follows: Labels, $j^{*(i)}$, are first assigned to all audio fragments, as in Equation (IV.7). Next, the constraints are incorporated while iterating through each fragment i. The log-likelihood, with constraints, of fragment i under each mixture component j is computed;

$$\log \tilde{p}(y^{(i)}; \Theta_j) = \log p(y^{(i)}; \Theta_j) - \sum_{c=-W/2, c\neq 0}^{W/2} \phi_j(j^{*(i+c)}), \tag{IV.8}$$

where W is the length of the temporal neighborhood surrounding the i^{th} fragment over which the constraints are imposed, and

$$\phi_j(j^{*(c)}) = \begin{cases} 0 & \text{if } j^{*(c)} = j \\ \lambda & \text{otherwise} \end{cases} \tag{IV.9}$$

adds a penalty of λ when the neighboring class labels $j^{*(c)}$ do not match the current label j. The new constrained class label of the fragment, $j^{*(i)}$, is then assigned according to:

$$j^{*(i)} = \underset{j}{\operatorname{argmax}} \log \tilde{p}(y^{(i)}; \Theta_j) \tag{IV.10}$$

The process is iterated, for all i, until convergence of the class labels for all fragments. The fixed cost parameter, λ, and the neighborhood size, W, over which the constraints are imposed are determined experimentally and depend on the type of feature and sequence step size being used. We find that $\lambda \approx 90$ and a constraint neighborhood corresponding to 15-20 seconds is optimal.

IV.E.4 Refining Segment Boundaries

This first segmentation is relatively coarse and can localize segment boundaries, at best, to within 0.25 sec, due to the large step size and the poor localization properties of using long audio fragments. Precise boundaries are found by extracting audio fragments with shorter length (∼1.75 sec) and step size (∼0.05 sec). We assign these short fragments to the *same* DTM components learned in Section IV.E.2, resulting in a finer segmentation of the song. This tends to over-segment songs as the DTM state changes too frequently: the coarse segmentation more accurately learns the temporal structure of each song. However, we can refine the original, coarse segmentation by moving each segment boundary to the closest corresponding boundary from the fine segmentation. These refined boundaries are likely to be valid since they were produced by the same DTM model. They are expected to provide a more precise estimate of the true segment boundaries.

IV.F Segmentation Evaluation

In this section, we evaluate the proposed algorithm for song segmentation on two music datasets. We also test the applicability of the algorithm to the similar task of music boundary detection.

IV.F.1 Data

We evaluate the automatic song segmentation performance of the DTM model on two separate musical datasets for which human-derived structural segmentations exist:

RWC dataset

The RWC Music Database (RWCMDB-P-2001) [47] contains 100 Japanese pop songs where each song has been segmented into coherent parts by a human listener [48]. The segments are accurate to 10ms and are labeled with great detail.

For this work we group the labeled segments into 4 possible classes: "verse" (i.e., including verse A, verse B, etc.), "chorus", "bridge" and "other" ("other" includes labels such as "intro", "ending", "pre-chorus", etc. and is also used to model any silent parts of the song). This results in a "ground truth" segmentation of each song with 4 possible segment classes. On average, each song contains 11 segments (with an average segment length of 18.3 seconds).

PopMusic dataset

The second dataset is a collection of 60 popular songs from multiple genres including rock, pop and hip-hop. Half the tracks are by the Beatles and the remainder are from a selection of popular artists from the past 40 years including Radiohead, Michael Jackson and the Beastie Boys. The human segmentations for this dataset were used by Levy and Sandler [65] to evaluate their musical segmentation algorithm. The ground truth segmentation of each song contains between 2 and 15 different segment classes (mean = 6.3) and, on average, each song also contains 11 segments (with an average segment length of 16.5 seconds).

IV.F.2 Experimental Setup

The songs in the RWC dataset were segmented with the DTM model into $K = 4$ segments (chosen to model "verse", "chorus", "bridge" and "other" segments and for comparison to previous work on the same dataset [112]) using the method described in Section IV.E. DTM models trained using either the MFCC or chroma features, we denote DTM-MFCC and DTM-Chroma, respectively. For DTM-MFCC, we use a sequence length of 900 MFCC feature vectors (extracted from 5.2 seconds of audio content) and a step-size of 100 feature frames, while for DTM-Chroma, we use a sequence length of 600 chroma feature vectors (13.9 seconds of audio) and a step-size of 20 frames. The dimension of the hidden state-space of the DTM was $n = 7$ for MFCC, and $n = 6$ for chroma.

For comparison, we also segment the songs using a Gaussian mixture

model (GMM) trained on the same feature data [5]. We learn a $K = 4$ component GMM for each song, and segment by assigning features to the most likely Gaussian component. Since segmentation decisions are now made at the short time-scale of individual feature vectors, we smooth the GMM segmentation with a length-1000 maximum-vote filter. We compare these models against two baselines: "constant" assigns all windows to a single segment, "random" selects segment labels for each window at random.

We quantitatively measure the correctness of a segmentation by comparing with the ground-truth using two clustering metrics: 1) the Rand index [56] intuitively corresponds to the probability that any pair of audio fragments will be clustered correctly, with respect to each other (i.e, in the same cluster, or in different clusters); 2) the pairwise F-measure [65] compares pairs of feature sequences that the model labels as belonging to the same segment-type with the true segmentation. If P_m is the set of audio fragment pairs that the model labels as similar and P_h is the set of fragment pairs that the human segmentation indicates should be similar then:

$$\begin{aligned} \text{pairwise precision, } P_{\text{pairwise}} &= \frac{|P_m \cap P_h|}{|P_m|} \\ \text{pairwise recall, } R_{\text{pairwise}} &= \frac{|P_m \cap P_h|}{|P_h|} \\ \text{pairwise F-measure} &= \frac{2 * P_{\text{pairwise}} * R_{\text{pairwise}}}{P_{\text{pairwise}} + R_{\text{pairwise}}} \end{aligned}$$

We also report the average number of segments per song.

IV.F.3 Segmentation Results

Table Table IV.1 reports the segmentation results on the RWC dataset. DTM-MFCC outperforms all other models, with a Rand index of 0.75[1] and a pairwise F-measure of 0.62. GMM performs significantly worse than DTM, e.g., the F-measure drops to 0.52 on the MFCC features. In particular, the GMM grossly

[1]This Rand index result is slightly lower than the value reported in [6] as, in the current work, we allow each model segment to match only one reference segment. This is consistent with the evaluations in [65].

Table IV.1 Song segmentation of the RWC dataset.

Model	Rand Ind	Pairwise F	Av. Segments
DTM-MFCC	0.75	0.62	10.8
DTM-Chroma	0.73	0.58	11.9
GMM-MFCC	0.66	0.52	58.7
GMM-Chroma	0.61	0.51	26.3
Constant	0.32	0.48	1
Random	0.57	0.32	279.0
Truth	1.00	1.00	11

over-segments the songs, leading to very low pairwise precision. This suggests that there is indeed a benefit in modeling the temporal dynamics with the DTM.

For the PopMusic dataset, we no longer restrict the segmentation to just 4 classes and instead attempt to model all possible segment classes. Given that each song in the dataset has an average of 6.3 different segments, we set the number of mixture components in the DTM model $K = 6$ and increase the state-space dimension to $n = 12$. The segmentation results are shown in Table Table IV.2 and are very similar to the results obtained for the RWC dataset. We note that the DTM-MFCC model F-measure of 0.6196 +/- 0.0163 improves on the segmentation algorithm of Levy and Sandler [65] who report an average F-measure of 0.603 using $K = 6$ clusters to segment the same data. The result of [65] lies at the minimum of our confidence interval. A paired comparison of the results for each song would be required to conclusively determine the significance of our improvement, but this data was not available in [65]. The state-of-the-art segmentation performance validates the DTM's capacity to model musical audio content and its promise for applications beyond segmentation, as a general, generative model for music.

Looking at the different feature representations, the DTM-MFCC outper-

Table IV.2 Song segmentation of the PopMusic dataset.

Model	Rand Ind.	Pairwise F	Av. Segments
DTM-MFCC	0.78	0.62	10.7
DTM-Chroma	0.74	0.51	12.0
GMM-MFCC	0.72	0.49	78.9
GMM-Chroma	0.67	0.50	32.4
Constant	0.32	0.48	1
Random	0.57	0.32	279.0
Truth	1.00	1.00	11.1

forms DTM-Chroma on both datasets, with F-scores of 0.62 vs 0.58 on RWC, and 0.62 vs 0.51 on PopMusic. On the other hand, GMM-MFCC and GMM-Chroma perform similarly (F-scores of 0.52 vs 0.51 on RWC, and 0.49 vs. 0.50 on Pop). These results suggest that chroma time-series are not as well modeled as MFCC time-series by the DTM model. In particular, each coordinate of the chroma feature vector is active (non-zero) when a particular musical key is present, and hence the time-series of chroma features will tend to be "spiky", depending on when the chords change in the song. The chroma features are also non-negative. Because of these two aspects, the chroma time-series is not as well modeled by the DTM, which is better suited for modeling second-order smooth time-series with Gaussian noise.

Table Table IV.3 examines the impact of the musical constraints and boundary refinement on the segmentations produced by our best model, the DTM-MFCC model. We see that the musical constraints improve the final segmentation of the PopMusic dataset by removing short, inaccurate segments and thus reducing the overall number of segments (the average number of segments drops from 17.9 to 10.7 where the true segmentations contain an average of 11.1 segments). Indeed,

Table IV.3 Effect of musical constraints and boundary refinement on DTM-MFCC segmentation of the PopMusic dataset.

Model	Rand Ind.	Pairwise F	Av. Segments
Coarse	0.762	0.577	17.9
Refine	0.770	0.587	17.9
Constrain	0.773	0.614	10.7
Constrain+Refine	0.779	0.620	10.7

these constraints often remove certain segment classes from the output altogether. In cases where the true segmentation had less than K different classes, the model can now ignore irrelevant classes.

Examples of DTM song segmentations are compared to the ground truth in Figures Figure IV.4 and Figure IV.5-Figure IV.8. We see that, while most DTM segments are accurate, there are a few errors due to imprecise borders, and some cases where the model over- or under-segments.

IV.F.4 Boundary Detection Results

In addition to evaluating the segmentation performance of the DTM model, we can consider its accuracy in detecting the boundaries between segments (without trying to label the segment classes). We evaluate boundary detection performance using two median time metrics: true-to-guess (T-to-G) and guess-to-true (G-to-T) respectively measure the median time from each true boundary to the closest model estimate, and the median time from each model estimate to the closest true boundary, as in [112]. We also consider the precision, recall and F-measure of boundary detection where a boundary output by the model is considered a "hit" if it is within a certain time threshold of a true segment boundary, as in [65, 81, 112].

The boundary detection results, averaged over the 100 RWC songs ($K = 4$),

are presented in Table Table IV.4. We use a threshold of 0.5 seconds for comparison to [112], who tackle the boundary detection problem by learning a supervised classifier that is optimized for boundary detection. In Table **??**, we show results for the PopMusic dataset ($K = 6$) where we now use a hit threshold of 3 seconds, following [65] and [81]. For both datasets, we also compare with the music analysis company EchoNest [1], which offers an online service for automatically detecting music boundaries.

Table IV.4 DTM boundary detection performance on the RWC dataset, compared to a commercial online service "the EchoNest" and the supervised method of [112].

Model	G-to-T	T-to-G	P	R	F
DTM-MFCC	3.21	2.96	0.22	0.22	0.22
DTM-Chroma	5.69	4.46	0.10	0.12	0.11
EchoNest	5.08	1.84	0.18	0.21	0.19
[112]	4.29	1.82	0.33	0.46	0.38

Table IV.5 DTM boundary detection performance on the PopMusic dataset compared to EchoNest.

Model	G-to-T	T-to-G	P	R	F
DTM-MFCC	3.58	2.99	0.62	0.65	0.61
DTM-Chroma	5.82	4.39	0.41	0.46	0.42
EchoNest	5.59	5.32	0.41	0.56	0.45

For the PopMusic dataset, the boundary detection results for the DTM segmentation (boundary F-measure = 0.61) are comparable to the performance of Levy and Sandler's segmentation algorithm (best boundary F-measure = 0.604) [65].

However, neither system approaches the accuracy of specialized boundary detection algorithms (e.g., Ong and Herrera [81] achieve boundary F-measure of 0.75 on a test set of similar Beatles music). Boundary detection algorithms (e.g., [81, 112]) are designed to detect novelty between successive feature frames or respond to musical cues such as drum fills or changes in instrumentation which indicate that one segment is ending and another beginning. However, they do not model the musical structure and there is no characterization of the segments between the boundaries as the DTM or [65] provides. In future work, we will investigate using a supervised boundary detection algorithm to improve on the simple refinement of the DTM segmentation that we propose in Section IV.E.4.

IV.G Applications of Automatic Song Segmentation

In this section, we demonstrate several applications of the automatic song segmentation algorithm to music annotation, retrieval, and visualization.

IV.G.1 Autotagging Song Segments

A number of algorithms have been proposed for automatically associating music content with descriptive semantic phrases or "tags" [38, 111, 125]. These supervised methods use large corpora of semantically tagged music to discover patterns in the audio content that are correlated with specific tags. Various methods exist for collecting the tags used to train these systems including hiring human subjects to label songs [111], mining websites [61], or online games [113] (see [110] for a review of the performance of each of these methods).

The tags generated by most of these method are presumed to be associated with the entire song. However, depending on the specific tag and the source from which it was collected, this may not be true. For example, the song "Bohemian Rhapsody" by Queen might accurately be tagged by one listener as a "melancholy piano ballad", another listener might refer to the "energetic opera with falsetto

vocal harmonies", while a third listener might hear "screaming classic rock with a powerful electric guitar riff". The training of autotagging algorithms [38, 111] is designed to accommodate the fact that not all of the features present in the labeled music audio content will actually manifest the associated tags. However, this "multiple instance learning" problem presents a challenge for evaluating the output of such algorithms since many of the tags apply to only certain segments of the song.

The solution to the "Bohemian Rhapsody problem" lies in first dividing a song into musically homogeneous segments and then tagging each of the segments individually. We use the music tagging algorithm described in [111] to associate the segments extracted from the 60-song PopMusic dataset described in Section IV.F with 149 semantic tags from the CAL-500 vocabulary used in [111]. Given a music waveform, the output of this algorithm is a *semantic multinomial distribution*, a vector of probabilities that each tag in the vocabulary applies to the music content. These tags include genre, emotion, instrument, vocal style and song-usage descriptors. The accuracy of the tagging algorithm has been found to predict one human's responses as accurately as another human would [111] (i.e., it approaches the limit imposed by musical subjectivity) and was the best performing automatic music tagging algorithm in the 2008 Music Information Retrieval Evaluation eXchange (MIREX) contest [9].

Figure Figure IV.9 demonstrates the $K = 6$ class DTM segmentation of the song "Bohemian Rhapsody". Four of the top automatically-determined tags are displayed for each segment where the first indicates the segment's most likely genre, the second detects the most prevalent instrument or vocal characteristic, the third describes the emotion evoked by the segment and the fourth gives a general description of the segment. The majority of the tags accurately describe the musical content although a few are clearly incorrect (e.g., there is no saxophone in the second segment and, though his voice was high pitched, Freddie Mercury was not a female singer!). More importantly, there is a big difference between the

tags that describe the mellow, acoustic, early segments of the song and those used to describe the more rocking, up-tempo segments towards the end. Compare the tags for each segment in Figure Figure IV.9 with the top tags output for the entire song which generically describe Bohemian Rhapsody as a *pop* song with a *female vocal* that is *pleasant* and is *not very danceable*.

Table Table IV.6 further illustrates the need for segmentation before semantic analysis of audio content. In the left column, we present the average Kullback-Leibler (KL) divergence between the semantic multinomial describing a single, automatically-extracted segment of a given song from the PopMusic dataset (e.g., the first chorus of song 1) and other segments from that song that are assigned to the same DTM component (e.g., other choruses from song 1), segments from the same song but different classes (e.g., verse, bridge, etc. from song 1) and segments chosen randomly from any other song in the dataset, averaged over all songs from the PopMusic dataset. This method of using semantic descriptors to determine audio similarity has been shown to be more accurate than calculating similarity of the acoustic content directly [7]. Table Table IV.6 demonstrates that while segments assigned to the same DTM components produce almost identical semantic descriptions (KL = 0.04), there is a large divergence between the semantic multinomial distributions of segments from different DTM components from within the same song (KL = 0.54), approaching the divergence between two random segments (KL = 0.70).

The right column of Table Table IV.6 presents the average tempo mismatch between segments, averaged over all songs from the PopMusic dataset. We use an automatic tempo extraction algorithm [33] to compute the tempo, in beats-per-minute (bpm), of each segment. As in [52], we deem two segments to have similar tempi if the bpm of the second is within ±4% of the bpm of the first, where, to account for confusion in the meter, matches with one-third, half, double or triple the first bpm are also permitted. We see that segments from the same class differ in tempo 20% of the time whereas two random segments have almost 50% chance

Table IV.6 Mean semantic KL divergence and tempo mismatch between a DTM segment and another segment from the same class, from the same song (but a different class) and from a different song, averaged over all songs from the PopMusic dataset. Section IV.G.B explains the Similar DT (bottom row).

	KL divergence	Tempo mismatch
Same Class	0.04	0.20
Same Song	0.54	0.29
Different Song	0.70	0.49
Similar DT	0.33	0.29

of a tempo mismatch. The average tempo mismatch between segments from the same class in the true segmentation is 10%. These results suggest that the DT is also capturing temporal information, along with the semantic information.

IV.G.2 Song Segment Retrieval

The automatic segmentation of a song and the representation of a segment as a series of audio fragments as a coherent similar dynamic texture can now be used to retrieve songs with musically similar segments, answering questions like "what sounds similar to the verse of this song?". We represent each segment by its corresponding dynamic texture component in the DTM-MFCC model and measure similarities between dynamic textures with the KL divergence between them [21](note that this KL divergence is now between dynamic texture models, rather than the KL between semantic multinomial distributions considered in the previous section and presented in Table Table IV.6). Using each song segment from the RWC dataset as a query, the five closest retrieved segments are presented online[2]. Qualitatively, the retrieved segments are similar in both audio texture

[2] http://cosmal.ucsd.edu/cal/projects/segment/

and temporal characteristics. For example, a segment with slow piano will retrieve other slow piano songs, whereas a rock song with piano will retrieve more upbeat segments.

To quantitatively evaluate the song segment retrieval, we compute the average semantic KL divergence and tempo mismatch between each query segment and the retrieved song segments that are modeled with the most similar dynamic texture component. The results for the single most similar DT are presented in the bottom row of Table Table IV.6. It can be seen that two segments with most similar DT components are, on average, more semantically similar than two segments from the same song (KL of 0.33 vs 0.54). The tempo mismatch between retrieved segments is the same as segments from the same song but significantly lower than segments from different songs (note that 75% of the most similar retrieved segments came from the same song as the query - DT components of the same DTM model). This indicates that the dynamic texture model captures both the timbre of the audio content, evidenced by the similar semantic descriptions (derived from analysis of the instantaneous spectral characteristics), as well as temporal characteristics, as shown by the similar tempi.

In order to visualize the distribution of songs in the dataset, the automatically extracted segments of songs from the PopMusic dataset were embedded into a 3-D manifold using local-linear embedding (LLE) [94] of the KL similarity matrix computed above for song retrieval. Two dimensions of the embedding are shown in Figure Figure IV.10.

We add interpretability to this embedding by inferring genre and emotion tags that best describe each part of the space. For each tag, we compute a kernel density estimate of the tag's probability distribution by placing a Gaussian kernel at each segment point in the embedding space. We weight each kernel by the tag probability assigned to the corresponding segment by the autotagging algorithm described in Section IV.G.1. The result is an estimate of the distribution over the embedding space of a each tag's relevance. In Figure Figure IV.10, we label the

embedding space by finding the centroid of the area of the top 20% of each of these probability densities.

The four emotion tags in Figure Figure IV.10 illustrate that the largest variance in the DTM segments results in good separation between the tags "happy" and "sad" and between "calming" and "arousing", corresponding with the psychological primitives or "core affect" described in [96]. The six genre tags show a progression from synthesized music like "hip hop" and "electronica" in the lower right, through "blues" and "pop" in the center to "rock" and "punk" at the top left. This automatic labeling of the embedding space again suggests that the DTM model is successfully capturing both the audio texture (e.g., separating happy and sad) and the temporal characteristics (e.g., separating calming and arousing) of the songs.

IV.H Conclusions

We have presented a new representation for musical audio, the dynamic texture (DT), which simultaneously accounts for both the instantaneous content of short audio fragments as well as the evolution of the audio over time. We applied the new representation to the task of song segmentation (i.e., automatically dividing a song into coherent segments that human listeners would label as verse, chorus, bridge, etc.), by modeling audio fragments from a song as samples from a dynamic texture mixture (DTM) model. Experimentally, the resulting segmentation algorithm achieves state-of-the-art results in segmentation experiments on two music datasets. More importantly, the generative nature of the proposed model of music makes it directly applicable to a wider and more diverse range of applications, compared to algorithms specifically developed for music segmentation. Its state-of-the-art results on music segmentation indicate that the dynamic texture representation shows promise as a new model for automatic music analysis. Future work will consider using the DTM model to move beyond the bag-of-features representation

in applications such as music similarity and automatic music tagging.

Another interesting direction for future work is to use more complex "switching" DT models [23, 24, 46] to improve on the DTM segmentation. These models should better localize the segment boundaries, as they operate on the entire song, rather than in a fragment-based manner. In general, these switching models are more difficult to learn robustly, due to the complexity of the models and the necessity for approximate inference. However, their effectiveness can be greatly increased by initializing the learning algorithm with a good segmentation, such as the one provided by the proposed DTM segmentation algorithm. Also a potential direction of future work is to modify the DTM so that it better models the properties of the chroma time-series.

IV.I Acknowledgements

This research utilized the AIST Annotation for the RWC Music Database (Popular Music Database) and the Queen Mary reference structural segmentations. We made use of the EchoNest developer web services at http://developer.echonest.com.

Chapter 4, in full, is a reprint of the material as it appears in IEEE Transactions on Audio, Speech and Language Processing 18(3). L. Barrington, A.B. Chan and G.R.G Lanckriet, 2010. The dissertation author was the primary investigator and author of this paper.

Algorithm 1 EM for a Mixture of Dynamic Textures

Input: N sequences $\{y^{(i)}\}_{i=1}^{N}$, number of components K.

Initialize $\Theta = \{\Theta_j, \alpha_j\}_{j=1}^{K}$.

repeat

{Expectation Step}

for $i = \{1, \ldots, N\}$ and $j = \{1, \ldots, K\}$ **do**

Compute the conditional expectations

$$\begin{aligned} \hat{x}_{t|j}^{(i)} &= \mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}\left[x_t^{(i)}\right], \\ \hat{P}_{t,t|j}^{(i)} &= \mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}\left[x_t^{(i)}(x_t^{(i)})^T\right], \\ \hat{P}_{t,t-1|j}^{(i)} &= \mathrm{E}_{x^{(i)}|y^{(i)},z^{(i)}=j}\left[x_t^{(i)}(x_{t-1}^{(i)})^T\right], \end{aligned}$$

by running Kalman smoothing filter with parameters Θ_j on sequence $y^{(i)}$.

Compute the posterior assignment probability

$$\begin{aligned} \hat{\mathbf{z}}_{i,j} &= p(z^{(i)} = j | y^{(i)}) \\ &= \frac{\alpha_j p(y^{(i)}|z^{(i)} = j)}{\sum_{k=1}^{K} \alpha_k p(y^{(i)}|z^{(i)} = k)}. \end{aligned}$$

end for

{Maximization Step}

for $j = 1$ to K **do**

Compute aggregate expectations

$$\begin{aligned} &\hat{N}_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j}, && \xi_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j}\hat{x}_{1|j}^{(i)}, && \Phi_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} \hat{P}_{t,t|j}^{(i)}, \\ &\varphi_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t|j}^{(i)}, && \eta_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j}\hat{P}_{1,1|j}^{(i)}, && \phi_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t-1,t-1|j}^{(i)}, \\ &\Psi_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t-1|j}^{(i)}, && \Lambda_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} (y_t^{(i)} - \bar{y}_j)(y_t^{(i)} - \bar{y}_j)^T, \\ &\Gamma_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} (y_t^{(i)} - \bar{y}_j)(\hat{x}_{t|j}^{(i)})^T, \\ &\gamma_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} y_t^{(i)}, && && \beta_j = \textstyle\sum_i \hat{\mathbf{z}}_{i,j} \sum_{t=1}^{\tau} \hat{x}_{t|j}^{(i)}. \end{aligned}$$

Compute new parameters $\{\Theta_j, \alpha_j\}$

$$\begin{aligned} &C_j^* = \Gamma_j(\Phi_j)^{-1}, && A_j^* = \Psi_j(\phi_j)^{-1}, && R_j^* = \tfrac{1}{\tau\hat{N}_j}\left(\Lambda_j - C_j^*\Gamma_j\right), \\ &Q_j^* = \tfrac{1}{(\tau-1)\hat{N}_j}\left(\varphi_j - A_j^*\Psi_j^T\right), && \mu_j^* = \tfrac{1}{\hat{N}_j}\xi_j, && S_j^* = \tfrac{1}{\hat{N}_j}\eta_j - \mu_j^*(\mu_j^*)^T, \\ &\bar{y}_j^* = \tfrac{1}{\tau\hat{N}_j}(\gamma_j - C_j^*\beta_j). && \alpha_j^* = \tfrac{\hat{N}_j}{N}, \end{aligned}$$

end for

until convergence

Output: $\{\Theta_j, \alpha_j\}_{j=1}^{K}$

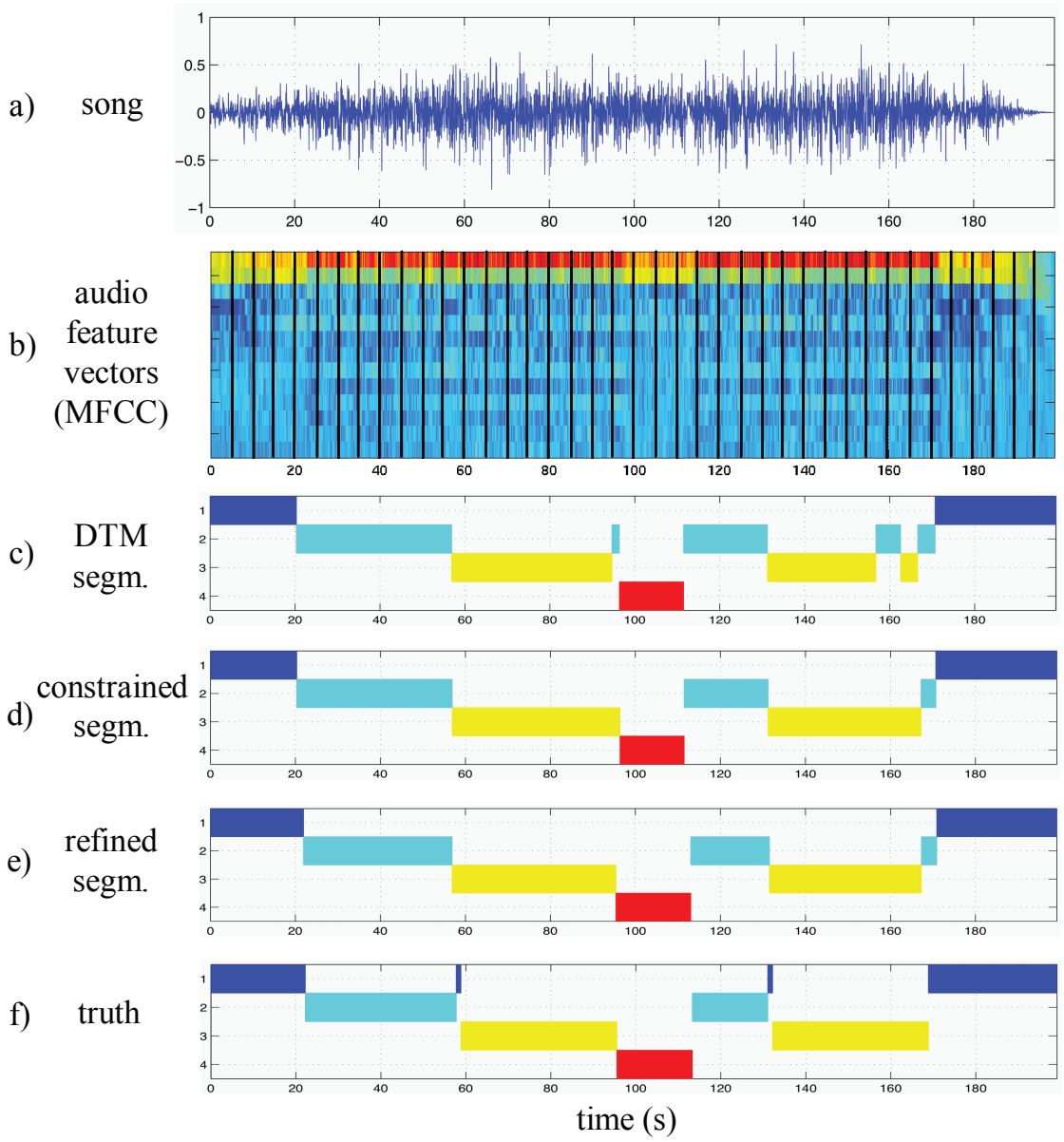


Figure IV.3 DTM song segmentation. A song's waveform (a) is represented as a series of audio feature vectors that are collected into short, overlapping sequences (b). These sequences of feature vectors are modeled as a dynamic texture mixture and the song is segmented based on the dynamic texture mixture component to which each sequence is assigned (c). Segments are constrained (d) and refined (e) to produce a final segmentation which is evaluated with reference to a human labeled ground-truth segmentation (f).

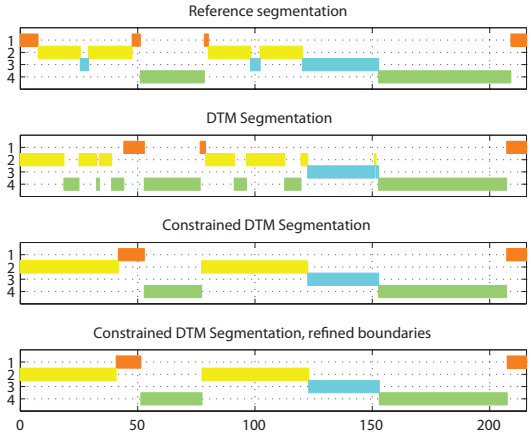


Figure IV.4 DTM segmentations and reference segmentation of the track "p053" from the RWC dataset (Rand Index = 0.78, Pairwise F = 0.66). The addition of the musical constraints removes short segments.

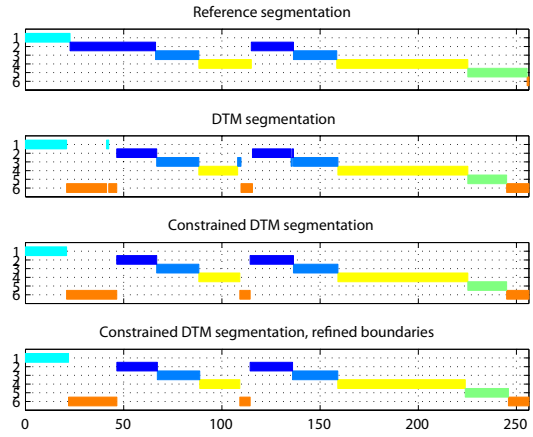


Figure IV.5 DTM segmentations and reference segmentation of 'Wonderwall' by Oasis. This is an example of an accurate segmentation where the DTM model captures almost all the reference segments but incorrectly divides the verse (class 2) into 2 parts (these in fact correspond to singing / no singing).

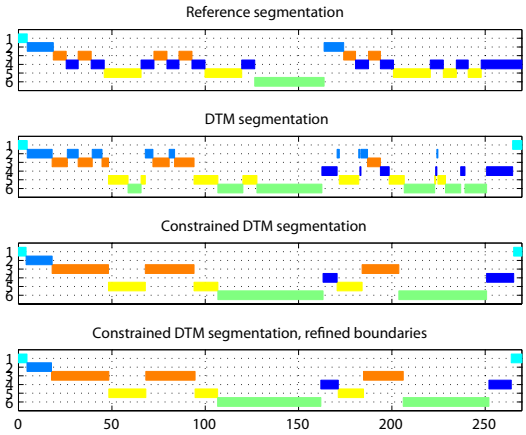


Figure IV.6 DTM segmentations and reference segmentation of 'Drive' by R.E.M. This is an example of a poor segmentation where the DTM model under-segments the "refrain" class (class 3) and the constraints incorrectly expand class 6.

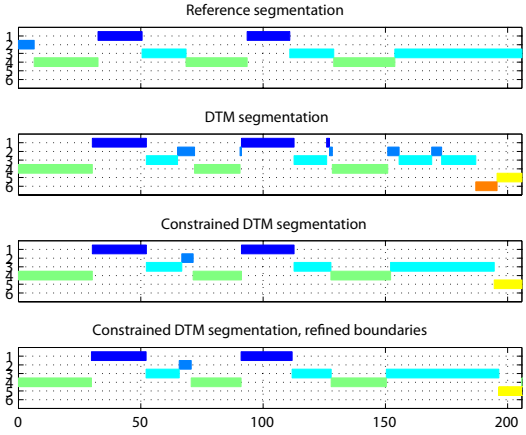


Figure IV.7 DTM segmentations and reference segmentation of 'Lucy In The Sky With Diamonds' by The Beatles. The addition of the musical constraints allows the DTM model to remove extra segment classes when there are more mixture components than necessary.

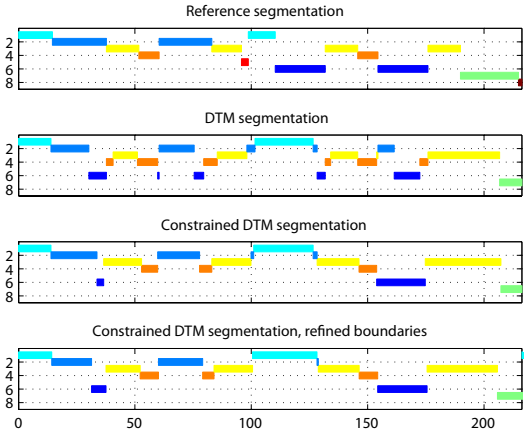


Figure IV.8 DTM segmentations and reference segmentation of 'It's Oh So Quiet' by Björk from the PopMusic dataset (Rand Index = 0.82, Pairwise F = 0.55). When there are more classes in the reference segmentation than there are DTM components, the model successfully ignores the smallest classes.

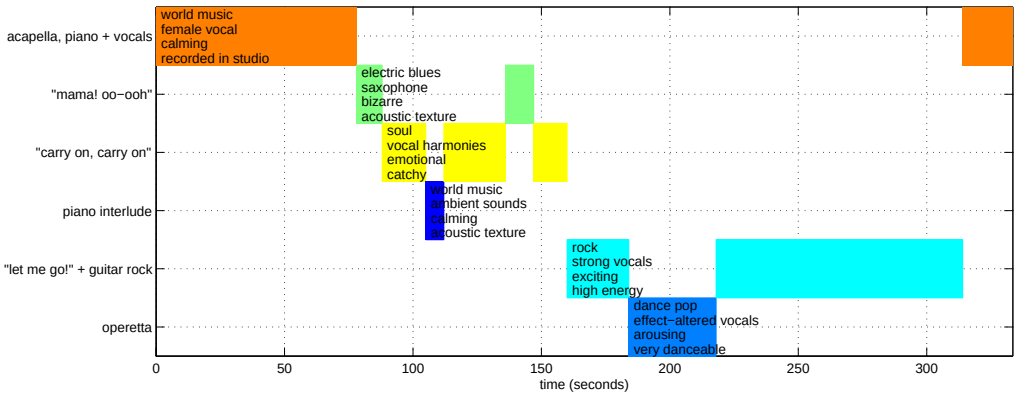


Figure IV.9 DTM segmentation of the song "Bohemian Rhapsody" by Queen. The automatically generated tags show the most likely genre, the most prevalent instrument or vocal characteristic, the emotion evoked and a general description of each segment class. Treating the song as a whole results in the general tags *pop*, *female vocal*, *pleasant* and *not very danceable*. The y-axis labels are added by the authors to highlight the musical or lyrical content of each segment class.

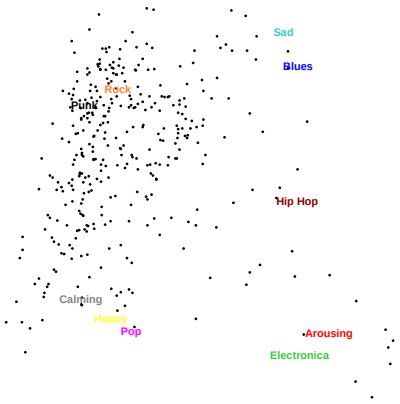


Figure IV.10 2-D visualization of the distribution of song segments. Each black dot is a song segment. Areas of the space are automatically tagged based on the system described in Section IV.G.1.

Chapter V

Conclusion

This thesis described machine learning and human crowdsourcing methods that have combined to advance the growing field of computer audition by analyzing, segmenting, describing, searching and recommending massive datasets of musical content. As the methods described here mature and evolve, they point towards the realization of machines that replicate a fundamentally human task and begin to understand music.

Thanks for listening.

Bibliography

[1] "EchoNest," http://the.echonest.com.

[2] S. Abdallah, M. Sandler, C. Rhodes, and M. Casey, "Using duration models to reduce fragmentation in audio segmentation," *Machine Learning: Special Issue on Machine Learning in and for Music*, vol. 65, no. 2-3, p. 485515, December 2006.

[3] G. T. Ajay Kapur, Manjinder Benning, "Query by beatboxing: Music information retrieval for the dj," *ISMIR*, 2004.

[4] V. Ambati, S. Vogel, and J. Carbonell, "Active learning and crowd-sourcing for machine translation," in *7th Conference on International Language Resources and Evaluation (LREC)*, 2010.

[5] J.-J. Aucouturier, F. Pachet, and M. Sandler, "'The Way It Sounds': Timbre models for analysis and retrieval of music signals," *IEEE Transactions on Multimedia*, vol. 7, no. 6, pp. 1028–1035, 2005.

[6] L. Barrington, A. B. Chan, and G. Lanckriet, "Dynamic texture models of music," in *IEEE ICASSP*, 2009, pp. 1589–1592.

[7] L. Barrington, A. Chan, D. Turnbull, and G. Lanckriet, "Audio information retrieval using semantic similarity," in *ICASSP*, 2007.

[8] L. Barrington, M. Yazdani, D. Turnbull, and G. Lanckriet, "Combining feature kernels for semantic music retrieval," in *9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[9] L. Barrington, D. Turnbull, and G. Lanckriet, "Auto-tagging music content with semantic multinomials," http://www.music-ir.org/mirex/2008/abs/AT_barrington.pdf, October 2008.

[10] M. Bartsch and G. Wakefield, "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001, pp. 15–19.

[11] D. Bauer, "Comparing the CCA subspace method to pseudo maximum likelihood methods in the case of no exogenous inputs," *Journal of Time Series Analysis*, vol. 26, pp. 631–68, 2005.

[12] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman, "A large-scale evalutation of acoustic and subjective music-similarity measures," *Computer Music Journal*, vol. 28, no. 2, pp. 63–76, 2004.

[13] J. Berg, R. Forsythe, F. Nelson, and T. Rietza, "Results from a dozen years of election futures markets research," *Handbook of Experimental Economics Results*, vol. 1, pp. 742–751, 2008.

[14] D. M. Blei and M. I. Jordan, "Modeling annotated data," *26th International Conference on Research and Development in Information Retrieval (ACM SIGIR)*, 2003.

[15] A. Brew, D. Greene, and P. Cunningham, "Using crowdsourcing and active learning to track sentiment in online media," in *6th Conference on Prestigious Applications of Intelligent Systems (PAIS), Lisbon, Portugal*, 2010.

[16] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 30, no. 1-7, pp. 107–117, 1998.

[17] C. R. Buchanan, "Semantic-based audio recognition and retrieval," Master's thesis, School of Informatics, University of Edinburgh, 2005.

[18] P. Cano and M. Koppenberger, "Automatic sound annotation," in *IEEE workshop on Machine Learning for Signal Processing*, 2004.

[19] G. Carneiro and N. Vasconcelos, "Formulating semantic image annotation as a supervised learning problem," *IEEE CVPR*, 2005.

[20] W. Chai and B. Vercoe, "Music thumbnailing via structural analysis," in *Proceedings of the 11th ACM international conference on Multimedia*, 2003, pp. 223–226.

[21] A. B. Chan and N. Vasconcelos, "Probabilistic kernels for the classification of auto-regressive visual processes," in *IEEE CVPR*, vol. 1, 2005, pp. 846–851.

[22] ——, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909–926, May 2008.

[23] ——, "Layered dynamic textures," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 10, pp. 1862–1879, 2009.

[24] ——, "Variational layered dynamic textures," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

[25] D. Cohn, Z. Ghahramani, and M. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Results*, vol. 4, no. 1, pp. 129–145, 1996.

[26] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popovic, and Foldit Players, "Predicting protein structures with a multiplayer online game," *Nature*, vol. 466, pp. 756–760, 2010.

[27] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley-Interscience, 1991.

[28] E. Coviello, L. Barrington, G. Lanckriet, and A. Chan, "Automatic music tagging with time series models," in *Eleventh International Symposium for Music Information Retrieval (ISMIR)*, 2010.

[29] V. Dani, O. Madani, D. Pennock, and S. Sanghai, "An empirical comparison of algorithms for aggregating expert predictions," in *22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

[30] R. B. Dannenberg and N. Hu, "Understanding search performance in query-by-humming systems," *ISMIR*, 2004.

[31] S. Dasgupta and L. Schulman, "A probabilistic analysis of em for mixtures of separated, spherical Gaussians," *Journal of Machine Learning Research*, vol. 8, pp. 203–226, 2007.

[32] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, pp. 1–38, 1977.

[33] S. Dixon, "Mirex 2006 audio beat tracking evaluation: Beatroot," *ISMIR*, 2006. [Online]. Available: http://www.ofai.at/~simon.dixon/beatroot

[34] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *Intl. J. Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

[35] G. Doretto, D. Cremers, P. Favaro, and S. Soatto, "Dynamic texture segmentation," in *IEEE ICCV*, vol. 2, 2003, pp. 1236–1242.

[36] J. S. Downie, "Music information retrieval evaluation exchange (mirex)," http://www.music-ir.org/mirex2006.

[37] ——, "Audio tag classification," Music Information Retrieval Evaluation eXchange (MIREX) http://music-ir.org/mirex/wiki/2008:Audio_Tag_Classification_Results, 2008.

[38] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green, "Automatic generation of social tags for music recommendation," in *21st Conference on Neural Information Processing Systems (NIPS)*, 2007.

[39] G. Eisenberg, J.-M. Batke, and T. Sikora, "Beatbank - an mpeg-7 compliant query by tapping system," *Audio Engineering Society Convention*, 2004.

[40] D. Ellis and G. Poliner, "Identifying cover songs with chroma features and dynamic programming beat tracking." IEEE ICASSP, 2007.

[41] S. Essid, G. Richard, and B. David, "Inferring efficient hierarchical taxonomies for music information retrieval tasks: Application to music instruments." *ISMIR*, 2005.

[42] S. L. Feng, R. Manmatha, and V. Lavrenko, "Multiple bernoulli relevance models for image and video annotation," *IEEE CVPR*, 2004.

[43] J. Foote, "Visualizing music and audio using self-similarity," in *International Multimedia Conference*, 1999, pp. 77 – 80.

[44] D. Forsyth and M. Fleck, "Body plans," *IEEE CVPR*, 1997.

[45] M. free MP3 music and music licensing, "Magnatune," http://www.magnatune.com.

[46] Z. Ghahramani and G. E. Hinton, "Variational learning for switching state-space models," *Neural Computation*, vol. 12, no. 4, pp. 831–864, 2000.

[47] M. Goto, "Development of the RWC music database," in *International Congress on Acoustics*, April 2004, pp. 553–556.

[48] ——, "AIST annotation for RWC music database," in *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, October 2006, pp. 359–360.

[49] ——, "A chorus selection detection method for musical audio singals and its application to a music listening station," *IEEE TASLP*, vol. 14-5, 2006.

[50] M. Goto and K. Hirata, "Recent studies on music information processing," *Acoustical Science and Technology*, vol. 25, no. 4, pp. 419–425, 2004.

[51] J. Gould and C. Lewis, "Designing for usability: key principles and what designers think," *Communications of ACM*, vol. 28, no. 3, pp. 300–311, 1985.

[52] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Speech and Audio Processing*, no. 5, pp. 1832–1844, 2006.

[53] M. Hoffman, D. Blei, and P. Cook, "Easy as CBA: A simple probabilistic model for tagging music," in *10th International Conference on Music Information Retrieval (ISMIR)*, 2009.

[54] J. Hsu, C. Liu, , and L. Chen, "Discovering nontrivial repeating patterns in music data," *IEEE Trans. on Multimedia*, vol. 3, no. 3, pp. 311–325, September 2001.

[55] X. Hu, J. S. Downie, and A. F. Ehmann, "Exploiting recommended usage metadata: Exploratory analyses," *ISMIR*, 2006.

[56] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[57] International Game Developers Association, "Casual games white paper," 2008. [Online]. Available: http://www.igda.org/casual/IGDA_Casual_Games_White_Paper_2008.pdf

[58] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory.* Prentice-Hall, 1993.

[59] Y. Kim, E. Schmidt, and L. Emelle, "Moodswings: A collaborative game for music mood label collection," in *9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[60] P. Knees, T. Pohle, M. Schedl, D. Schnitzer, and K. Seyerlehner, "A document-centered approach to a natural language music search engine," in *30th European Conference on Information Retrieval (ECIR)*, 2008.

[61] P. Knees, T. Pohle, M. Schedl, and G. Widmer, "A music search engine built upon audio-based and web-based similarity measures," in *30th International Conference on Research and Development in Information Retrieval (ACM SIGIR)*, 2007.

[62] W. E. Larimore, "Canonical variate analysis in identification, filtering, and adaptive control," in *IEEE Conf. on Decision and Control*, vol. 2, 1990, pp. 596–604.

[63] E. Law and L. von Ahn, "Input-agreement: A new mechanism for collecting data using human computation games," in *27th International Conference on Human Factors in Computing Systems (ACM CHI)*, 2009.

[64] E. Law, K. West, M. Mandel, M. Bay, and S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *10th International Conference on Music Information Retrieval (ISMIR)*, 2009.

[65] M. Levy and M. Sandler, "Structural segmentation of musical audio by constrained clustering," *IEEE Trans. on Acoustics, Speech and Language Processing*, vol. 16, no. 2, pp. 318–326, February 2008.

[66] M. Levy, M. Sandler, and M. Casey, "Extraction of high-level musical structure from audio data and its application to thumbnail generation," in *IEEE ICASSP*, 2006.

[67] D. Lewis and J. Catlett, "Heterogeneous Uncertainty Sampling for Supervised Learning," in *11th International Conference on Machine Learning*, 1994, pp. 148–156.

[68] T. Li and G. Tzanetakis, "Factors in automatic musical genre classification of audio signals," *IEEE WASPAA*, 2003.

[69] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *1st International Conference on Music Information Retrieval (ISMIR)*, 2000.

[70] B. Logan and S. Chu, "Music summarization using key phrases," in *IEEE ICASSP*, 2000, pp. 749–752.

[71] A. Makadia, V. Pavlovic, and S. Kumar, "A new baseline for image annotation," in *10th European Conference on Computer Vision (ECCV)*, 2008.

[72] M. Mandel and D. Ellis, "Multiple-instance learning for music information retrieval," in *9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[73] ——, "A web-based game for collecting music metadata," *Journal of New Music Research*, vol. 37, no. 2, pp. 151–165, June 2008.

[74] B. McFee and G. Lanckriet, "Heterogeneous embedding for subjective artist similarity," in *Tenth International Symposium for Music Information Retrieval (ISMIR2009))*, October 2009.

[75] I. McGraw, A. Gruenstein, and A. Sutherland, "A self-labeling speech corpus: Collecting spoken words with an online educational game," in *INTERSPEECH*, 2009.

[76] M. McKinney and J. Breebaart, "Features for audio and music classification," in *ISMIR*, 2003.

[77] S. Ness, A. Theocharis, G. Tzanetakis, and L. Martins, "Improving automatic music tag annotation using stacked generalization of probabilistic SVM outputs," in *17th ACM International Conference on Multimedia (ACM MM)*, 2009.

[78] S. Novotney and C. Callison-Burch, "Cheap, fast and good enough: Automatic speech recognition with non-expert transcription," in *Human Language Technologies: 11th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, 2010.

[79] S. Nowak and S. Rüger, "How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation," in *11th ACM International Conference on Multimedia Information Retrieval (ACM MIR)*, 2010.

[80] B. Ong, E. Gómez, and S. Streich, "Automatic extraction of musical structure using pitch class distribution features," in *Workshop on Learning the Semantics of Audio Signals*, 2006.

[81] B. Ong and P. Herrera, "Semantic segmentation of music audio contents," in *ISMIR*, 2005.

[82] P. V. Overschee and B. D. Moor, "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, pp. 75–93, 1994.

[83] F. Pachet and D. Cazaly, "A taxonomy of musical genres," *RIAO*, 2000.

[84] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *6th International Conference on Music Information Retrieval (ISMIR)*, 2005.

[85] E. Pampalk, "Computational models of music similarity and their application to music information retrieval," Ph.D. dissertation, Vienna University of Technology, 2006.

[86] A. Patel, J. Iversen, M. Bregman, and I. Schulz, "Experimental evidence for synchronization to a musical beat in a nonhuman animal," *Current Biology*, vol. 19, no. 10, pp. 827–830, 2009.

[87] J. Paulus and A. Klapuri, "Music structure analysis using a probabilistic fitness measure and an integrated musicological model," in *ISMIR*, 2008.

[88] G. Peeters, A. Burthe, and X. Rodet, "Toward automatic music audio summary generation from signal analysis," in *Proceedings of the 3rd Conference on Music Information Retrieval (ISMIR)*, 2002, pp. 94–100.

[89] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[90] V. C. Raykar, S. Yu, L. Zhao, G. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, vol. 11, pp. 1297–1322, April 2010.

[91] J. Reed and C. Lee, "A study on music genre classification based on universal acoustic models," in *7th International Conference on Music Information Retrieval (ISMIR)*, 2006.

[92] P. Rentfrow and S. Gosling, "Message in a ballad: The role of music preferences in interpersonal perception," *Psychological Science*, vol. 17, no. 3, pp. 236–242, 2006.

[93] D. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[94] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[95] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: a database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 3, pp. 157–173, 2008.

[96] J. Russell, "Core affect and the psychological construction of emotion," *Psychological Review*, vol. 110, no. 1, pp. 145–172, 2003.

[97] P. Saisan, G. Doretto, Y. Wu, and S. Soatto, "Dynamic texture recognition," in *IEEE CVPR*, vol. 2, 2001, pp. 58–63.

[98] S.Cherry, "Bet on it," *Spectrum, IEEE*, vol. 44, no. 9, pp. 48–53, 2007.

[99] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2010.

[100] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[101] M. Slaney, "Mixtures of probability experts for audio retrieval and indexing," *IEEE Multimedia and Expo*, 2002.

[102] ——, "Semantic-audio retrieval," *27th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.

[103] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng, "Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks," in *13th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

[104] A. Sorokin and D. Forsyth, "Utility data annotation with Amazon Mechanical Turk," in *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2008.

[105] J. Surowiecki, *The Wisdom of Crowds*. Anchor, 2004.

[106] D. Tingle, Y. Kim, and D. Turnbull, "Exploring automatic music annotation with "acoustically-objective" tags," in *IEEE International Conference on Multimedia Information Retrieval (MIR)*, 2010.

[107] D. Torres, D. Turnbull, L. Barrington, and G. Lanckriet, "Identifying words that are musically meaningful," in *ISMIR*, 2007.

[108] T. Tullis and B. Albert, *Measuring the User Experience: Collecting, Analyzing and Presenting Usability Metrics*. Morgan Kaufmann, 2008.

[109] D. Turnbull, L. Barrington, and G. Lanckriet, "Modelling music and words using a multi-class naïve bayes approach," in *ISMIR*, 2006.

[110] ——, "Five approaches to collecting tags for music," in *ISMIR*, 2008.

[111] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Trans. on Acoustics, Speech and Language Processing*, vol. 16, no. 2, pp. 467–476, February 2008.

[112] D. Turnbull, G. Lanckriet, E. Pampalk, and M. Goto, "A supervised approach for detecting boundaries in music using difference features and boosting," in *Proceedings of the 8th Conference on Music Information Retrieval (ISMIR)*, 2007.

[113] D. Turnbull, R. Liu, L. Barrington, D. Torres, and G. Lanckriet, "Using games to collect semantic information about music," in *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[114] G. Tzanetakis and P. R. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 7 2002.

[115] N. Vasconcelos, "Image indexing with mixture hierarchies," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[116] S. Vijayanarasimhan and K. Grauman, "Cost-sensitive active visual category learning," *International Journal of Computer Vision*, vol. 91, pp. 24–44, 2011.

[117] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2004.

[118] T. Volkmer, J. Smit, and A. Natsev, "A web-based system for collaborative annotation of large image and video collections: an evaluation and user study," in *13th ACM International Conference on Multimedia (ACM MM)*, 2005.

[119] L. von Ahn, "Games with a purpose," *IEEE Computer Magazine*, vol. 39, no. 6, pp. 92–94, 2006.

[120] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *22nd International Conference on Human Factors in Computing Systems (ACM CHI)*, 2004.

[121] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-Based Character Recognition via Web Security Measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/321/5895/1465

[122] P. Welinder, S. Branson, S. Belongie, and P. Perona, "The multidimensional wisdom of crowds," in *Neural Information Processing Systems (NIPS)*, 2010.

[123] J. Whitehill, P. Ruvolo, J. Bergsma, T. Wu, and J. Movellan, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *23rd Conference on Neural Information Processing Systems (NIPS)*, 2009.

[124] B. Whitman, "Learning the meaning of music," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[125] B. Whitman and D. Ellis, "Automatic record reviews." in *5th International Conference on Music Information Retrieval (ISMIR)*, 2004.

[126] B. Whitman and R. Rifkin, "Musical query-bydescription as a multiclass learning problem," in *IEEE Multimedia Signal Processing Conference (MMSP)*, December 2002.