Contextual tag inference

MICHAEL I. MANDEL, RAZVAN PASCANU, DOUGLAS ECK, and YOSHUA BENGIO, Université de Montréal
LUCA M. AIELLO and ROSSANO SCHIFANELLA, Università di Torino
and FILIPPO MENCZER, Indiana University

This paper examines the use of two kinds of context to improve the results of content-based music taggers: the relationships between tags and between the clips of songs that are tagged. We show that users agree more on tags applied to clips temporally "closer" to one another; that conditional restricted Boltzmann machine models of tags can more accurately predict related tags when they take context into account; and that when training data is "smoothed" using context, support vector machines can better rank these clips according to the original, unsmoothed tags and do this more accurately than three standard multi-label classifiers.

Categories and Subject Descriptors: H.1.2 [**Information Systems**]: Models and Principles—*Human information processing*; H.5.5 [**Information Interfaces and Presentation**]: Sound and Music Computing—*Signal analysis, synthesis, and processing*; I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and neural nets*

General Terms: Experimentation, Performance

Additional Key Words and Phrases: Autotagging, clips, context, music, smoothing, tags

ACM Reference Format:

ACM Trans. Multimedia Comput. Commun. Appl. V, N, Article A (September 10), 17 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The amount of digital music available online is vast and increasing, not just as official releases on record labels, but also as user generated content. Social tags and expert generated tag descriptions allow users to find relevant content on Last.fm and Pandora.com, respectively, but they are time consuming and expensive to collect. This paper discusses methods for automatically applying tags to music ("autotagging") to scale these data beyond the limitations of current manual processes.

Automatically generated tags can be used in three primary ways. They can be used to search by description through a collection of music that has been automatically described. For example, a user might want to find music that is "folk rock with male vocals and acoustic guitar." They can be used to browse through music with similar descriptions. For example, the same search could be constructed for a user automatically from David Bowie's "Space Oddity," without having to articulate a specific description.

Author's address: M. Mandel: University of Montreal, Department of Computer Science and Operations Research, CP 6128, Succ. Centre-Ville, Montréal, Québec, H3C 3J7 CANADA

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 10 ACM 1551-6857/10/09-ARTA $10.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Finally, they allow summarization of search results and other collections of music. Descriptions of minutes or hours of music can be skimmed in seconds from which relevant candidates can be selected for more thorough auditory screening.

This paper specifically discusses ways of taking *context* into account for improving autotagger training data. Context in this case means some combination of the relationships between tags and the relationships between 10-second clips of songs. Autotagging systems typically treat each tag as an independent classification problem, and each clip of music as an independent data point. The models described in this work take advantage of the relationships between tags and between clips to make better predictions of tags.

After discussing previous work on autotagging and context in machine vision in Section 1.1, Section 2 describes measurements of temporal context in an existing dataset. Section 2.1 then describes what is to our knowledge the first use of Amazon's Mechanical Turk service to collect autotagging training data, and another examination of temporal context on these data. Section 3 then describes the various language models employed to capture tag-tag contextual relationships, specifically, the restricted Boltzmann machine (RBM), the conditional RBM (CRBM), and the doubly conditional RBM (DCRBM), and information theoretic models of tag-tag similarity. Finally, Section 4 describes four experiments conducted with these models on three different datasets, measuring their prediction of the tags themselves and the improvement that they impart to support vector machine-based autotaggers.

1.1 Background

Researchers have investigated a number of music autotagging techniques over the last decade [Slaney 2002; Whitman and Rifkin 2002; Eck et al. 2008; Tingle et al. 2010]. Many current autotagging systems (e.g. [Mandel and Ellis 2008]) treat each tag as a separate classification or ranking problem. This paper generalizes this to the problem of predicting the presence or relevance of multiple tags simultaneously, which is known as multi-label classification [Tsoumakas et al. 2010]. Many multi-label classification algorithms have been proposed in the literature [Boutell et al. 2004; Kang et al. 2006; Tsoumakas and Vlahavas 2007; Zhang and Zhou 2007; Han et al. 2010], including some explicitly designed for problems in music [Trohidis et al. 2008].

One form of context that we explore in this paper is the exploitation of the relationships between tags. This form of context has been explored by other authors with regards to music tagging, generally as a second stage on top of independent tag predictors. Aucouturier et al. [2007] use a decision tree learned from ground truth tags to enforce certain relationships between tag predictions and find that it improves the precision of their taggers by an average of 5% and up to 15%. Bertin-Mahieux et al. [2008] use a second layer of boosting on top of the individual autotaggers, but do not show a significant improvement over the individual autotaggers. Miotto et al. [2010] use a Dirichlet mixture model to smooth autotagger outputs, improving the performance of acoustic classifiers and outperforming a support vector machine context model under a number of metrics.

The context model of Miotto et al. [2010] is inspired by context models from the vision literature, specifically, Rasiwasia and Vasconcelos [2009]. The vision and image retrieval literature in general has developed rather sophisticated notions of context that could be applied to music, such as Chen et al. [2010]. Heitz and Koller [2008] characterize these different types of context well, describing their own work as "stuff-thing" context, meaning that it uses background textures to inform its predictions of the identities of foreground objects. This is contrasted against the "scene-thing" context of Murphy et al. [2004], which first uses the "gist" of the image to classify the scene (e.g. office, hallway, street), and then uses the scene predictions to inform priors on where and what objects should be expected. Rabinovich et al. [2007] use a "thing-thing" context, where the identities of objects in scenes inform one another. This is perhaps the most similar of the three to the context used in music tagging. Other

visual contexts include the 3D geometric relationships between objects [Hoiem et al. 2008] and the relationship between human pose and objects like sports equipment [Yao and Fei-Fei 2010]. In both of these cases, object identification and context mutually reinforce each other.

These different types of visual context could have analogs in music. For example, a visual scene might be analogous to a musical genre, as the priors over instruments, moods, etc. found in a song should depend on the genre of the song. Just as an office scene has a high probability of including a computer keyboard [Murphy et al. 2004], a rock song has a high probability of involving guitar. Similarly, just as a street scene has a low prior on that same keyboard, a hip-hop song has a low prior on that same guitar. In the same way, spatial context in images could correspond to temporal context in music and the 3D geometric context of Hoiem et al. [2008] might correspond to a more nuanced notion of musical structural context. Just as Hoiem et al. [2008] infer the 3D geometry of a scene from a 2D projection of it and use it to perform better spatial reasoning about car and pedestrian detection, so might musical structure be inferred from the temporal and musical surfaces and used to better inform tagging. This paper will, however, restrict itself to a simpler notion of temporal context, defined simply according to clip metadata: its offset into a track, the track, the album, and the artist. In fact, many of our models will only consider the clip and track identities. While this is greatly simplified, it is still useful in modeling the temporal context of music and tags.

Our examination of the relationships between the tags of different parts of the same song was enabled by our collection of a new dataset using Amazon.com's Mechanical Turk service. Mechanical Turk is a marketplace where "requesters" post tasks that require some amount of basic human intelligence along with a bounty for their completion. For example, we paid "turkers" $0.03-0.05 per clip that they annotated with 5-10 tags. This marketplace allowed us to gather a new dataset relatively quickly and cheaply using 5 clips from each song. Mechanical Turk has been used in the past for collecting natural language processing data [Snow et al. 2008] and vision data [Sorokin and Forsyth 2008; Whitehill et al. 2009], but to our knowledge it has not been used for collecting tag data for music. Concurrently with our work, Lee [2010] used Mechanical Turk to collect music similarity judgements.

This paper is based on work presented in our previous conference publication [Mandel et al. 2010] and uses techniques described by Schifanella et al. [2010]. It applies the information theory-based smoothing methods of the latter with the autotaggers of the former, and adds to this a new RBM-based smoothing model. It also unifies the experiments presented in those papers and explicitly states the ideas of context in music.

2. TEMPORAL TAG CONTEXT

How similar are the tags that different users apply to the same clip? Different clips from the same track? Clips from different tracks on the same album? Clips from different albums from the same artist? Clips from different artists? How does this similarity vary for clips from the same track as the separation between the clips increases? We consider these questions and attempt to answer them through a simple co-occurrence analysis of tags similar to the one performed by Schifanella et al. [2010]. This investigation of temporal context or the temporal "scale" of various tags and types of tags will inform our tag language models.

Our existing MajorMiner dataset [Mandel and Ellis 2008] provides an excellent testbed for examining many of these questions. In order to perform this analysis, we measured the number of co-occurring tags in every pair of (user,clip) observations. We categorized each pair of observations by the relationships between the users and the clips. For example, they could be from the same user, and the clips could be from different tracks on the same album. Or they could be from different users but the same clip. We then simply took the average number of co-occurring clips for each of these various categorizations. Figure 1(a) shows the results of this analysis.

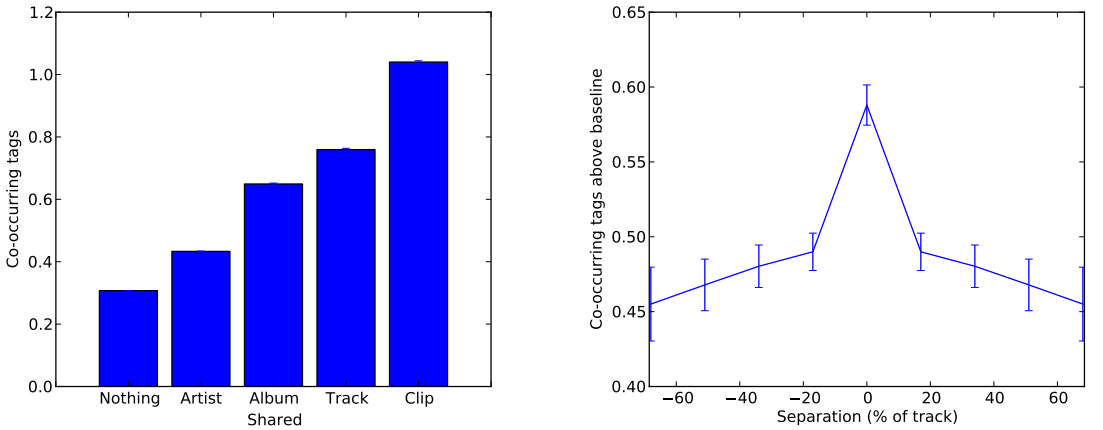


Fig. 1. (a) Average number of tags shared by human intelligence tasks (HITs) from different users as a function of the "distance" between the clips involved. (b) Average number of tags above the baseline shared by HITs from the same track as a function of the separation between the clips measured as % of a track.

It can be seen from this plot that listeners describe clips that are "closer" together with the same words more frequently. In this case, "closer" is in terms of the most specific scale of artists, albums, tracks, and clips which two clips share. Imagine grouping all of the clips in a collection into tracks, then all tracks into albums, then all albums into artists, and finally the artists in an arbitrary order. It is in this arrangement that closer clips are described more similarly. Of course, this does not take into account the ordering within any of these groupings, just the number of groupings that are shared.

Figure 1(a) quantifies the amount of noise that is introduced by assuming that tags at a certain scale apply to all of its associated clips, the finest scale. For example, assuming that tags applied to an artist apply equally well to all of the clips of music that the artist has released (as is done commonly, e.g. [Bertin-Mahieux et al. 2008]) implies that up to 50% noise is being introduced in those tags. This is because tags applied to clips from the same artist but different albums agree with each other only 50% as frequently as tags applied to the same clip. Similarly, assuming that tags applied to a track apply equally well to all of its clips means that up to 20% noise is being introduced.

2.1 Mechanical Turk data collection

While our existing MajorMiner dataset [Mandel and Ellis 2008] is sufficient for calculating these co-occurrences down to the clip level, it lacks sufficient numbers of clips from the same track to answer questions about the effect of separation within a single track. In order to investigate this property, we needed to collect a new dataset, and instead of attempting to attract a new batch of players to the MajorMiner game, we decided to pay users on Amazon's Mechanical Turk,[1] a marketplace for work the requires human intelligence.

The work of describing the music collection was broken down into individual human intelligence tasks (HITs), where in each HIT, users of the Mechanical Turk website were asked to listen to a single clip from a song and describe its unique characteristics using between 5 and 15 words. The task was free response, but to provide some guidance, we requested tags in 5 categories: Styles/Genres, Vocals/Instruments, Overall sound/feel (global qualities like production and rhythm), Moods/Emotions,

[1] `http://mturk.com`

and Other (sound alike artists, era, locale, song section, audience, activities, etc.). In order to avoid biasing the turkers' responses, no examples of tags were provided. Turkers were paid between $0.03 and $0.05 per clip, on which they generally spent about one minute.

The music used in the experiment was collected from music blogs that are indexed by the Hype Machine.[2] We downloaded the front page of each of the approximately 2000 blogs and recorded the URLs of any mp3 files linked from them, a total of approximately 17,000 mp3s. We downloaded 1500 of these mp3s at random, of which approximately 700 were available, error free, and at least 128 kbps while still being below 10 megabytes (to avoid DJ sets, podcasts, etc). Of these, we selected 185 at random. From each of these 185 tracks, we extracted five 10-second clips evenly spaced throughout the track. We presented these clips to turkers in a random order, and generally multiple clips from the same track were not available simultaneously. Each clip was seen by 3 different turkers.

Mechanical Turk gives the "requester" the opportunity to accept or reject completed HITs either manually or automatically. In order to avoid spammers, we designed a number of rules for automatically rejecting HITs based on analyses of each and all of a user's HITs. Individual HITs were rejected if: (1) they had fewer than 5 tags, (2) a tag had more than 25 characters, or (3) less than half of the tags were found in a dictionary of Last.fm tags. All of a users' HITs were rejected if: (1) that user had a very small vocabulary compared to the number of HITs they performed (fewer than 1 unique tag per HIT), (2) they used any tag too frequently (4 tags were used in more than half of their HITs), (3) they used more than 15% "stop words" like **nice**, **music**, **genre**, etc., or (4) at least half of their HITs were rejected for other reasons. The list of stop words was assembled by hand from HITs that were deemed to be spam.

We posted a total of 925 clips, each of which was to be seen by 3 turkers for a total of 2775 HITs. We accepted 2566 completed HITs and rejected 305 HITs. Some of the rejected HITs were re-posted and others were never completed. The completed HITs included 15,500 (user, tag, clip) triples from 209 unique turkers who provided 2100 unique tags. Of these tags, 113 were used by at least 10 turkers, making up 13,000 of the (user, tag, clip) triples. We paid approximately $100 for these data, although this number doesn't include additional rounds of data collection and questionnaire tuning.

2.1.1 *Results.* The data from Mechanical Turk provide an in-depth look at the relationships between tags applied to clips in the same track. Figure 1(b) shows the normalized number of tag co-occurrences for clips with different separations (in terms of percentage of the track). The normalization consists of calculating the difference between the number of tags that co-occur for clips at a particular offset in the *same* track versus those in *different* tracks. This normalization is necessary because different offsets into tracks were generally heard by different turkers with their own idiosyncratic vocabularies. With this normalization, it can be seen that there is a monotonic fall-off of tag co-occurrence from a peak at a 0% offset, supporting the data in Figure 1(a).

3. TAG LANGUAGE MODELS

Tags collected from human users are typically quite sparse. With Mechanical Turk, we were able to collect tags for a small number of clips, each seen by 3 people with 17 (user, tag) pairs on average. With the MajorMiner game, we were able to collect tags for a slightly larger number of clips, each seen by about 7 people with 30 (user, tag) pairs on average. The Last.fm dataset that we are using (see Section 4.1) has over 1.2 million tracks and 8.5 million (user, tag, track) triples, making an average of only 7 (user, tag) pairs per track. Additionally, because popular songs are tagged much more frequently than less popular songs, only 18% of songs have more than 7 (user, tag) pairs applied to them.

[2] `http://hypem.com/list`

For the popular tracks, it is not a problem to count the number of times each tag has been applied to each clip, but for those 82% of tracks in the long tail, these tag data are quite noisy. One approach to removing this noise is by treating the true tags as hidden data and inferring them from the observations of tags that were applied. Such a tag language model should be able to identify both tags that should have been applied, but weren't, and tags that were applied but should not have been.

This paper discusses and tests two different kinds of tag language models, one based on an information theoretic formulation of this inference [Schifanella et al. 2010], and the second based on restricted Boltzmann machines (RBMs) [Mandel et al. 2010; Mandel et al. 2011]. While there are many more uses for it, in this case, we only use the information theoretic model to learn the relationships between tags in a tag-tag similarity matrix. The RBM models can learn this tag-tag context, and can additionally incorporate the temporal context discussed in Section 2 through conditioning on "auxiliary variables."

3.1 Information theoretic models

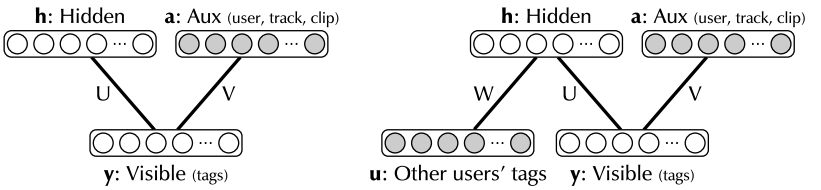
Inferring suitable hidden tags for *resources* (clips, tracks, albums and so on) with poor tagging information and detecting noisy tags in the set of tags attached to a resource are tasks that can be achieved by mining the global tag-to-tag similarity relationships occurring in the folksonomy.

Given a similarity score for every pair of tags, any set of tags associated with a specific resource can be expanded by adding the tags that are most similar to the given tags, under the assumption that tags which are strongly related to each other apply equally to the same resource. For instance, if **rap** and **hip-hop** are found to be very similar and one resource is tagged with only **rap**, it is very likely that **hip-hop** can be applied as well. Conversely, a noisy tag set can be reduced by wiping out those tags that are not strongly related to the majority of other tags assigned to the same resource. Here, we report a brief overview on how tag-to-tag similarity can be computed.

A widely accepted representation of folksonomies is based on a set of *triples* (u, r, t) representing a user u marking a resource r with a tag t. Extracting meaningful similarity patterns occurring between tags in such three-dimensional model requires a dimensionality reduction of the triple space, because measures for relatedness and similarity are still not well developed for three-mode data. The process of reducing a folksonomy to a two-mode relationship is known as *aggregation* [Markines et al. 2009]. Although the aggregation could be performed across any of the three dimension involved, in our setting we are interested in calculating the similarity between pairs of tags by means of comparing the resources they annotate; for this reason we aggregate on the user dimension. Doing so we obtain a description of each tag as a weighted vector of resources.

As widely shown by previous work [Markines et al. 2009], the aggregation can be performed in various ways, leading to as many different calculations of the weights in the resource vector. Simple examples of aggregation are the *projection* aggregation, which outputs a binary vector denoting the set of resources labeled by the tag, and the *distributional* aggregation, which produces a frequency-weighted vector where resources are weighted according to how many times they have been tagged with the considered tag.

Collapsing the two-dimensional tag-resource space into a weighted tag-to-tag relation is performed by applying a semantic similarity measure (e.g., cosine similarity) to each pair of resource vectors. Previous work addresses the application of different metrics in such context [Markines et al. 2009; Schifanella et al. 2010]. Since most of these metrics leverage fundamental principles from the information theory, in the following we will name *information theoretic* (or also *InfTh* for brevity) all the autotagging models that use these metrics.



(a) Conditional RBM (b) Doubly-conditional RBM

Fig. 2. Schematic diagrams of the two restricted Boltzmann machines under investigation. (a) RBM for tag smoothing conditioned on just auxiliary information: user, track, clips identity, (b) RBM for tag smoothing conditioned on auxiliary information and tags of other users. Filled circles show variables that are always observed, open circles show variables that are inferred at test time.

3.2 Restricted Boltzmann machines

This section describes the restricted Boltzmann machine (RBM) [Smolensky 1986] and conditional variants. The RBM is an undirected graphical model that generatively models a set of input variables $\mathbf{y} = (y_1, \ldots, y_C)^T$ with a set of hidden variables $\mathbf{h} = (h_1, \ldots, h_n)^T$. Both $\mathbf{y}$ and $\mathbf{h}$ are typically binary, although other distributions are possible. The model is "restricted" in that the dependency between the hidden and visible variables is bipartite, meaning that the hidden variables are independent when conditioned on the visible variables and vice versa. The joint probability density function is

$$p(\mathbf{y}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{y},\mathbf{h})} \text{ where } E(\mathbf{y}, \mathbf{h}) = -\mathbf{h}^T U \mathbf{y} - c^T \mathbf{h} - d^T \mathbf{y} \text{ and } Z = \sum_{\mathbf{y},\mathbf{h}} e^{-E(\mathbf{y},\mathbf{h})}. \quad (1)$$

The computation of Z, known as the partition function, is intractable because it is exponential either in the number of visible or hidden variables. The marginal of $\mathbf{y}$ is $p(\mathbf{y}) = e^{-\mathcal{F}(\mathbf{y})}/Z$, where $\mathcal{F}(\mathbf{y})$, the free energy of $\mathbf{y}$, is easy to compute as

$$\mathcal{F}(\mathbf{y}) = -\log \sum_{\mathbf{h}} e^{-E(\mathbf{y},\mathbf{h})} = -d^T \mathbf{y} - \sum_i \log(1 + e^{c_i + U_i \mathbf{y}}). \quad (2)$$

The parameters of the model can be optimized using gradient descent to minimize the negative log likelihood of data $\{\mathbf{y}_t\}$ under this model

$$\frac{\partial}{\partial \theta} p(\mathbf{y}_t) = -\mathbb{E}_{\mathbf{h} \mid \mathbf{y}_t} \left[\frac{\partial}{\partial \theta} E(\mathbf{y}_t, \mathbf{h}) \right] + \mathbb{E}_{\mathbf{y},\mathbf{h}} \left[\frac{\partial}{\partial \theta} E(\mathbf{y}, \mathbf{h}) \right]. \quad (3)$$

The first expectation in this expression is easy to compute, but the second is intractable and must be approximated. One popular approximation for it is contrastive divergence [Hinton 2002], which uses a small number of Gibbs sampling steps *starting from the observed example* to sample from $p(\mathbf{y}, \mathbf{h})$.

RBMs can be conditioned on other variables [Taylor et al. 2007]. In general, as shown in Figure 2(b), both the hidden and visible units can be conditioned on other variables, $\mathbf{u} = (v_1, \ldots, v_d)^T$ and $\mathbf{a} = (a_1, \ldots, a_A)^T$, respectively. We refer to this model as the doubly-conditional RBM because of these two conditioning variables. Including these interactions, the energy function becomes

$$E(\mathbf{y}, \mathbf{h}, \mathbf{u}, \mathbf{a}) = -\mathbf{h}^T U \mathbf{y} - \mathbf{h}^T W \mathbf{u} - \mathbf{y}^T V \mathbf{a} - d^T \mathbf{y} - c^T \mathbf{h} \quad (4)$$

and $p(\mathbf{y}, \mathbf{h} \mid \mathbf{u}, \mathbf{a}) \propto e^{-E(\mathbf{y},\mathbf{h},\mathbf{u},\mathbf{a})}$. The vectors $V\mathbf{a}$ and $W\mathbf{u}$ act like additional biases on $\mathbf{y}$ and $\mathbf{h}$. By setting to 0 the appropriate W or V matrix or $\mathbf{u}$ or $\mathbf{a}$ vector, the conditioning can apply to only the visible units, as in Figure 2(a), which we will refer to as the conditional RBM in the rest of the paper, or only the hidden units (not shown). For an observed data point $\mathbf{y}_t, \mathbf{u}_t, \mathbf{a}_t$, the gradient of the log likelihood with

respect to a parameter θ becomes

$$\frac{\partial}{\partial\theta}\log p(\mathbf{y}_t \mid \mathbf{u}_t, \mathbf{a}_t) = -\mathbb{E}_{\mathbf{h}\mid\mathbf{y}_t,\mathbf{u}_t,\mathbf{a}_t}\left[\frac{\partial}{\partial\theta}E(\mathbf{y}_t,\mathbf{h},\mathbf{u}_t,\mathbf{a}_t)\right] + \mathbb{E}_{\mathbf{y},\mathbf{h}\mid\mathbf{u}_t,\mathbf{a}_t}\left[\frac{\partial}{\partial\theta}E(\mathbf{y},\mathbf{h},\mathbf{u}_t,\mathbf{a}_t)\right]. \quad (5)$$

Salakhutdinov et al. [2007] describe a conditional RBM used for collaborative filtering in which only the hidden variables are conditioned on other variables. Mandel et al. [2010] describe a conditional RBM used for modeling tags of the form of Figure 2(a), while this paper is the first to describe a doubly conditional RBM used for modeling tags.

3.3 Conditional RBM tag language model

As shown in Figure 2, both the conditional RBM and doubly-conditional RBM (DCRBM) can be used as tag language models. Both of them use the same variables and same representations for tags, hidden representations, and auxiliary information including user, track, and clip identity. The DCRBM additionally includes a representation of the tags that other users have applied to a particular clip.

The visible units in each RBM, $\mathbf{y}$, represent the tags that have been applied to a particular clip by a particular user in the form of a binary vector. These visible units are connected to the hidden units, $\mathbf{h}$, and units representing auxiliary information about the user and clip involved, $\mathbf{a}$. Specifically, each user is given their own auxiliary unit that indicates when it is that user applying the tags. This is a so-called "one-hot" representation of user, as only one user unit is ever 1 at a tim. Additionally, however, there are analogous representations in $\mathbf{a}$ for the clip that has been tagged and the track from which it came. In this way, for example, taggings of the same clip by different users can share information, as can taggings of different clips from the same track, and taggings of different clips by the same user. In the case of the DCRBM, the $\mathbf{u}$ vector is the average of all of the other binary tag vectors that apply to the same clip, providing the context of the tags other users have apply to the clip. The $\mathbf{u}$ vector is the same length as the $\mathbf{y}$ vector, but the elements of $\mathbf{u}$ are continuous values between 0 and 1, while the elements of $\mathbf{y}$ are strictly 0 or 1.

The various parameters learned by the model can capture specialized information about the relationships between these variables. In particular, this can be encouraged by applying an L_1 or L_2 prior to the W and V matrices during learning. When W and V are encouraged to be sparse, the biases d capture the overall frequency of each tag, U captures information about the correlations between tags independent of context, and W captures the relationships between tags in other taggings of the same clip. The V matrix captures any deviations in relative frequency between tags due to context, for example if a user tends to favor certain tags over others or if a particular clip tends to be described with tags that would normally not coincide.

After a CRBM is trained, it is used to estimate $p(\mathbf{y} \mid \mathbf{a})$, which provides the smoothing. This is done by drawing samples from the distribution for each particular setting of $\mathbf{a}$, i.e. for each clip. Because of its probabilistic formulation, the CRBM allows unobserved variables to be either imputed or marginalized away. In particular, it is desirable to provide smoothed tags independent of a particular user's biases, so the user variables in $\mathbf{a}$ are marginalized away. Additionally, the DCRBM can be provided with an average of all of the tag vectors for a particular clip in $\mathbf{u}$, from which it estimates $p(\mathbf{y} \mid \mathbf{a}, \mathbf{u})$.

4. EXPERIMENTS

This section describes a number of experiments performed to investigate the usefulness of the various aspects of contextual tag language models. It first describes the datasets on which the models are tested, then the features used to describe the audio, then the experiments themselves.

4.1 Datasets

Three different datasets were employed in these experiments. They came from Mechanical Turk, the MajorMiner game, and a crawl of Last.fm, roughly corresponding to small, medium, and large sizes.

The Mechanical Turk dataset is described in Section 2.1. This dataset had 925 clips, 2100 tags, 209 users, and 15,500 (user, clip, tag) triples. While it is the smallest dataset, the clips that compose it were selected in such a way as to give the best picture of within-track tag variation. Because of its size, we treated all (user, clip, tag) triples as true for the purposes of evaluation, even those that were not verified by two users.

The medium-sized dataset comes from the MajorMiner game [Mandel and Ellis 2008]. It has 2600 clips, 6700 tags, 600 users, and 80,000 (user, clip, tag) triples. This dataset gives a nice balance between size, diversity of music, and different scales of tagging. Its tags were collected at the clip level, and so are very specific. Because approximately 7 users saw each clip, we accept as true (clip, tag) pairs on which two different users agree. If only one user uses a (clip, tag) pair, we count it as an intermediate state, neither true or false, for evaluation purposes.

The largest dataset comes from the website Last.fm. Schifanella et al. [2010] describe the design of a crawler to gather the social tags describing individual tracks in Last.fm. While this is more fine-grained tag data than is usually collected from Last.fm (e.g. [Bertin-Mahieux et al. 2008]), it is still not at the clip level like the other two datasets. In order to evaluate clip-level classification scores, we assumed that track-level tags applied to all of a track's clips. In practice, we found that there was enough data to simply select the 10-second clip from the exact center of each track to use as a representative of the entire track. While the entire dataset has 1.2 million clips, 280,000 tags, 84,000 users, and 8.6 million (user, clip, tag) triples, many of these only appear infrequently. Limiting the dataset to clips, tags, and users that appear in 100 or more triples still retains 8900 users, 9400 tracks, 7100 tags, and 1.4 million triples.

We pre-processed the data by transforming tags into a canonical form. We normalized the spelling of decades and the word "and," removed phrases such as "sounds like" from the beginning of tags, removed words like "music," "sound," and "feel" from the ends of tags, and removed punctuation. We also *stemmed* each word in the tag so that different forms of the same word would match each other, e.g. **drums**, **drum**, and **drumming**.

4.2 Features

We use the audio features described by Mandel and Ellis [2008], which characterize the audio's timbre and rhythm. The features are all calculated over 10-second clips of songs. The timbral features are the mean and unwrapped covariance matrix of 18-dimensional Mel frequency cepstral coefficients (MFCCs). This feature captures information about music's production and instrumentation. It specifically ignores musical features like harmony and melody.

The rhythmic features could be called "envelope cepstrum." The spectrogram is divided into a number of frequency bands, and the FFT across time is taken of each band, yielding the modulation of each band. The low frequency modulations are kept (up to 10 Hz), the log magnitude is taken, and then the DCT along time is taken, yielding the "envelope cepstrum" in the different frequency bands. These bands are then stacked on top of one another to create a 200-dimensional rhythmic feature vector. These features capture information about the music's beat, tempo, and rhythm, attempting to separate the various signals from the drum kit: bass drum, snare, and hi-hat cymbal.

Both of these features are computed over all of the clips in the dataset and then normalized so that each dimension has zero mean and unit variance. Each datapoint's feature vector is then normalized again so that it has unit norm. This second normalization minimizes the number of outliers caused

by the heavy tails of the feature dimension distributions, preventing them from dominating distance calculations.

4.3 Experiment 1: predicting tags from context

This experiment investigates the predictive power of the various language models for tags and the advantages that context provides for these models. It is purely textual in that it does not involve the audio of the music at all, just the tags, tagging information, user information, and music metadata.

All three of the datasets described in Section 4.1 can be used in a leave-one-out tag prediction task. In this task, the relative probability of a novel observation is compared to that of the same observation with one bit flipped (one tag added or deleted). If the model has captured important structure in the data, then it will judge the true observation to be more likely than the bit-flipped version of it. This ratio is directly connected to the so-called pseudo-likelihood of the test set [Besag 1975]. Because it is a ratio of probabilities, it does not require the computation of the partition function, Z, which is very computationally intensive. Mathematically, the pseudo-likelihood is defined as

$$\mathrm{PL}(v \mid a) \equiv \prod_i p(v_i \mid v_{\setminus i}, a) = \prod_i \frac{p(v \mid a)}{p(v \mid a) + p(\tilde{v}_i \mid a)} \tag{6}$$

where v_i is the ith visible unit, $v_{\setminus i}$ is all of the visible units except for the ith unit, and $\tilde{v}_i$ is the observation v with the ith bit flipped. Even though our observation vectors are generally very sparse ($\sim$4% of the bits were 1s), the 1s are more important than the 0s, so we compute the average log pseudo-likelihood over the 1s and 0s separately and then average those two numbers together. This provides a better indication of whether the model can properly account for the tags that are present, and diminishes the importance of the tags that aren't present.

This leave-one-out tag prediction can be done with any model that computes the likelihood of tags. Thus we can train models with different combinations of auxiliary variables, or different structures entirely, as long as they can predict the likelihood of novel data. A baseline comparison to all of our RBMs is a factored model that estimates the probability of each tag independently from training data and then measures the likelihood of each tag independently on test data. Because of the independence of the variables, in this case the pseudo-likelihood is identical to the true likelihood.

We performed this experiment with the textual component of these three datasets, dividing the data 60-20-20 into training, validation, and test sets. The observations were shuffled, but then rearranged slightly to ensure that all of the auxiliary classes appeared at least once in the training set to avoid "out-of-vocabulary" problems. This experiment only used the singly-conditional RBM for clarity, as the doubly-conditional RBM has very different pseudo-likelihoods due to its being conditioned on tag data. We ran a grid search over the number of hidden units, the learning rate, and the regularization coefficients using only the track-based auxiliary variables, those with the most even coverage. This grid search involved training approximately 500 different models, each taking 10 minutes on average. We selected the system with the best hyperparameters based on the pseudo-likelihood of the validation dataset. Once we had selected reasonable hyperparameters, we ran experiments using all combinations of the auxiliary variables with the other hyperparameters held constant. Five different random divisions of the data allowed the computation of standard errors.

The log pseudo-likelihoods of the test datasets under these systems are shown in Table I. The results are not strictly comparable across datasets because they involved slightly different numbers of visible units. The results are shown on a per-bit basis, however, to facilitate comparison. These results show first that non-conditional restricted Boltzmann machines (rows with three $-$s) are much more effective than the factored models at modeling test data. This is because in addition to modeling the relative frequencies of tags, the RBM models the relationships between tags through its hidden units.

Table I. Average per-bit log pseudo-likelihood (less negative is better) for restricted Boltzmann machines conditioned on different types of auxiliary information. A + indicates that the auxiliary information was present, a − indicates that it was absent. The baseline system is a factored model evaluated in the same way.

	Auxiliary info			
Dataset	User	Track	Clip	$\log(\text{PL}) \pm \text{stderr}$
MajorMiner	+	+	+	-0.9179 ± 0.0088
MajorMiner	+	+	−	-0.9189 ± 0.0070
MajorMiner	+	−	−	-0.9416 ± 0.0074
MajorMiner	−	−	−	-1.0431 ± 0.0095
MajorMiner		baseline		-1.4029 ± 0.0024
Mech. Turk	+	+	−	-0.893 ± 0.015
Mech. Turk	+	−	−	-0.904 ± 0.013
Mech. Turk	+	+	+	-0.914 ± 0.012
Mech. Turk	−	−	−	-1.039 ± 0.013
Mech. Turk		baseline		-1.300 ± 0.007

Conditioning the RBM on auxiliary information (rows with at least one +) further improves the pseudo-likelihoods. Specifically, it seems that the most useful auxiliary variable is the identity of the user, but the identity of the track helps as well. Including clip information is slightly detrimental, although not statistically significantly so, possibly because it introduces a large number of extra parameters to estimate in the W_a matrix from few observations.

4.4 Experiment 2: information theoretic smoothing

It is possible to exploit the tag-tag similarity relationship derived from the information theoretic model defined in Section 3.1 to improve the autotagger's prediction accuracy through a process of remodulation of the tag scores that we call *smoothing*. The idea behind this kind of smoothing is that the weighted tag vectors can be remodulated through a redistribution of scores to increase the weight of the most relevant tags and to minimize those of the noisy ones.

The smoothing procedure can be reasonably applied in pre-processing on the tag vectors used in the learning set as well as in post-processing on the autotagger output. Pre-smoothing is aimed at refining the autotagger input to improve learning when the considered clips are poorly or noisily tagged, while post-smoothing is used to adjust the tag prediction to highlight the most relevant tags or to bring out relevant tags that have not been detected by the autotagger. The same smoothing algorithm is used in both pre-processing and post-processing phases, modulo the normalization of the tag vectors; in the following we outline the detail of the main smoothing steps.

Given a tag vector of weights $W = [w_1, ..., w_M]$ corresponding to tags $T = \{t_1, ..., t_M\}$, the first step of the smoothing is to scale down the distribution of scores by a *scale factor* $\alpha \in [0, 1]$, simply recurring to the scalar product $(1 - \alpha) \cdot W = W'$. The overall weight $w_\alpha = \sum_{i=1}^{M} \alpha \cdot w_i$ that is deducted from the score vector is then redistributed among those tags that are found to be similar to the existing tags, according to the tag-tag similarity relationship. Accordingly to the techniques described in Section 3.1, a tag-tag similarity matrix is calculated for all the pairs among the N most popular tags that mark all the clips in the dataset except the clip for which the autotagging is being made. For each tag $t_i \in T$ we extract its $K \leq N$ most similar tags $T_{sim}(t_i) = \{\tau_1, ..., \tau_K\}$, together with their corresponding similarity values $W_{sim}(t_i) = [w_1^\tau, ..., w_K^\tau]$. The contributions of the similarity vectors computed for all the tags are then

combined in a single *smoothing vector* S by means of a weighted vector sum where the contribution from each tag t_i is proportional to its original tag score w_i:

$$S = \sum_{i=1}^{M} w_i \cdot W_{sim}(t_i). \tag{7}$$

Depending on the set of tags in the $T_{sim}(t_i)$ vectors, the number of entries in the smoothing vector can go up to $K \times M$, where M is the number of original tags associated with the considered clip. The weight w_α is then distributed among the tags in S, proportionally to their smoothing score; such contributions are added to the scaled tag weight vector W', restoring its original overall weight and thus leading to the final result of the smoothing.

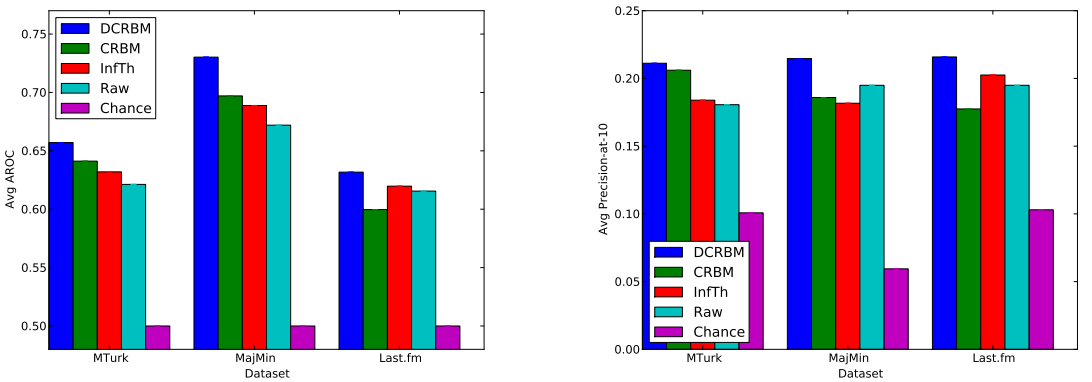
The redistribution of weights performed by the smoothing can (1) introduce new related tags in the original tag vector (2) increase the score of tags which are very related to many other tags given by the users and (3) decrease the score of tags which are unrelated, on average, to the others. Note that this process can result in a change of the weight ranking of the original tag score vector W.

For all the datasets we computed the tag-tag similarity matrix using the MIP similarity metric using a distributional aggregation (see [Markines et al. 2009] for details), which appears to be the best performing setting. We explored the effects of smoothing in both pre- and post-processing, for a wide range of N, K, and α parameters. For the smallest datasets, MajorMiner and Mechanical Turk, we found that the optimal smoothing parameters are $N = 100, K = 25, \alpha = 0.2$, while for the Last.fm dataset the best results are obtained using a bigger similarity matrix ($N = 200$) and assigning lower values to the scale factor and the number of related tags ($\alpha = 0.1, K = 10$). These changes can be interpreted by considering that the Last.fm folksonomy has a broader variety of distinct tags if compared to the other datasets; for this reason, restricting the similarity matrix size to less than 200 entries leaves out many tags which are relevant for the smoothing. Conversely, reduced α and K values avoid giving too much weight to noisy correlations which can be found inside the K-neighborhood of single tags in the tag-tag similarity matrix. Intuitively, noisy correlations are more frequent in a folksonomy extracted from a popular social network rather than in smaller triples sets created through on-purpose games (MajorMiner) or by paid workers (Mechanical Turk).

We observe that, for every dataset, our information theoretic model with pre-processing smoothing improves the accuracy of the baseline autotagger (see Figure 3). Surprisingly, the post-processing smoothing, which is not included in that plot, does not provide any accuracy improvement. This counterintuitive result could be due to the different distributions of weights of tags and autotags, that may lead to a ineffective weights redistribution during post-processing. However, no clear-cut and verifiable pattern that can shed light on the reason of this outcome has emerged from the analysis of the datasets. In the next Section, the information theoretic smoothing performance is compared with that of the other algorithms across the different datasets.

4.5 Experiment 3: smoothing on multiple datasets

This experiment compares the various smoothing algorithms on the various datasets. This comparison is in terms of the area under the ROC curve (AUC) and precision-at-10 of rankings of the raw, user-generated tags achieved by SVM classifiers trained using the smoothed tags. Specifically, for a given smoothing algorithm and dataset, we generate a smoothed version of the tags for each clip using a pre-trained instance of the algorithm with parameters selected to maximize the pseudo-likelihood of held-out tags. Then for each tag, we train a support vector machine (SVM) using the same number of positive and negative examples, namely the minimum of the number of positive examples, the number of negative examples, and 100 examples. The SVM is then evaluated on all of the test examples and



(a) (b)

Fig. 3. (a) Area under the ROC curve and (b) Precision-at-10 averaged across tags for support vector machines trained on raw and smoothed data and tested on the raw data.

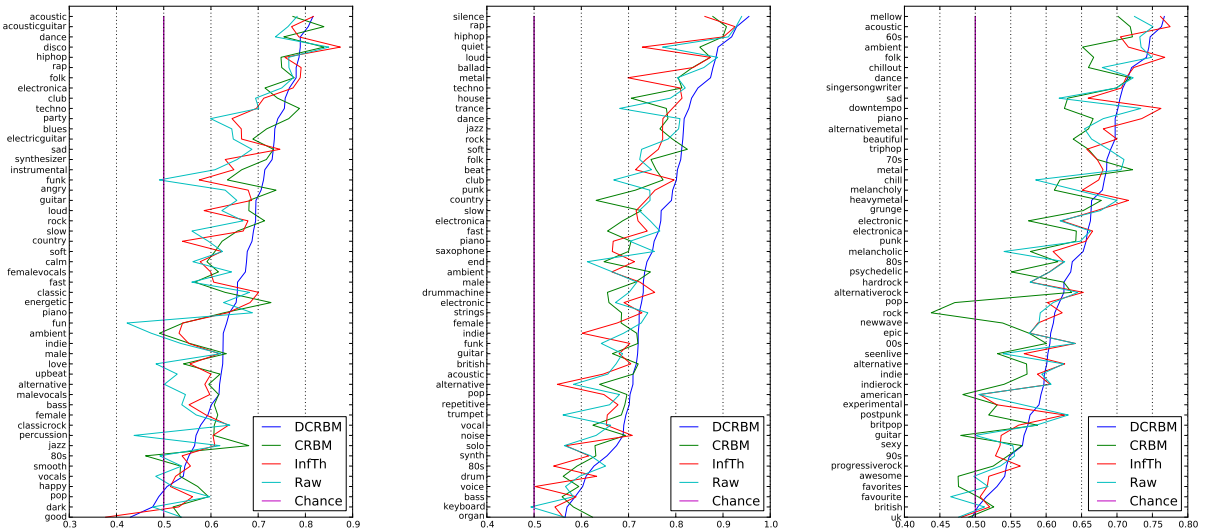
the distance from the hyperplane used to rank them. This is repeated for each tag and the results are averaged across tags. In this experiment, the various datasets used different numbers of tags. The Mechanical Turk dataset used 66 tags, the MajorMiner dataset used 71 tags, and the Last.fm dataset used 87 tags.

The AUC [Cortes and Mohri 2004] is a measure of ranking performance that summarizes the ability of retrieval system to rank positive examples above negative examples, scoring them on a scale from 0.5 (chance level) to 1 (perfect ranking). Although certain problems with it have been pointed out by Hand [Hand 2009], it is still a very popular and widely reported metric. Precision-at-10 is the proportion of positive examples in the top 10 results returned by a retrieval system, ranging between the baseline percentage of positive examples and 1. It is widely used in information retrieval, but although it addresses the AUC problems reported by Hand, it too has shortcomings, such as low stability and dependence on the total number of positive examples [Manning et al. 2008].

Results for this experiment can be seen in Figure 3. Almost all of the smoothings improve classification performance. This is a relatively surprising result. After training on these modified tag datasets, classifiers perform better when evaluated on the *original* dataset. Even though these datasets are intentionally mismatched in this way, the smoothing still proves to be beneficial.

Relative performance between the algorithms is consistent across datasets, with the doubly-conditional RBM performing the best, followed by the conditional RBM, the information-theoretic smoothing, and finally the raw user-supplied tags. Of the datasets, all algorithms perform better on the MajorMiner dataset than the other two. While this is clear from the AUC scores directly, for precision-at-10 the baseline for MajorMiner is lower, meaning that the various classifiers score relatively better above it than on the Mechanical Turk and Last.fm datasets. Two exceptions to these trends are the performance of the CRBM on the Last.fm dataset, which is worse than expected under both metrics and the precision-at-10 of the SVMs trained on raw tags on the MajorMiner dataset, which is better than expected. Because of the similarity in AUC and precision-at-10 results, and better stability of AUC, we focus on AUC results for the rest of the experiments.

The Mechanical Turk dataset, having the fewest number of tags applied to each clip, provides the least amount of certainty for each tagging. Thus these data are less reliable for training autotaggers and have the lowest performance. The MajorMiner dataset has more tags per clip, averaging out inaccurate tags and making agreement between users more apparent. Although the Last.fm dataset is the largest,



(a) Mechanical Turk (b) MajorMiner (c) Last.fm

Fig. 4. Accuracy of autotaggers for the top 50 tags in the Mechanical Turk, MajorMiner, and Last.fm datasets. The autotaggers were trained on raw and smoothed tags and tested on the raw, human generated tags.

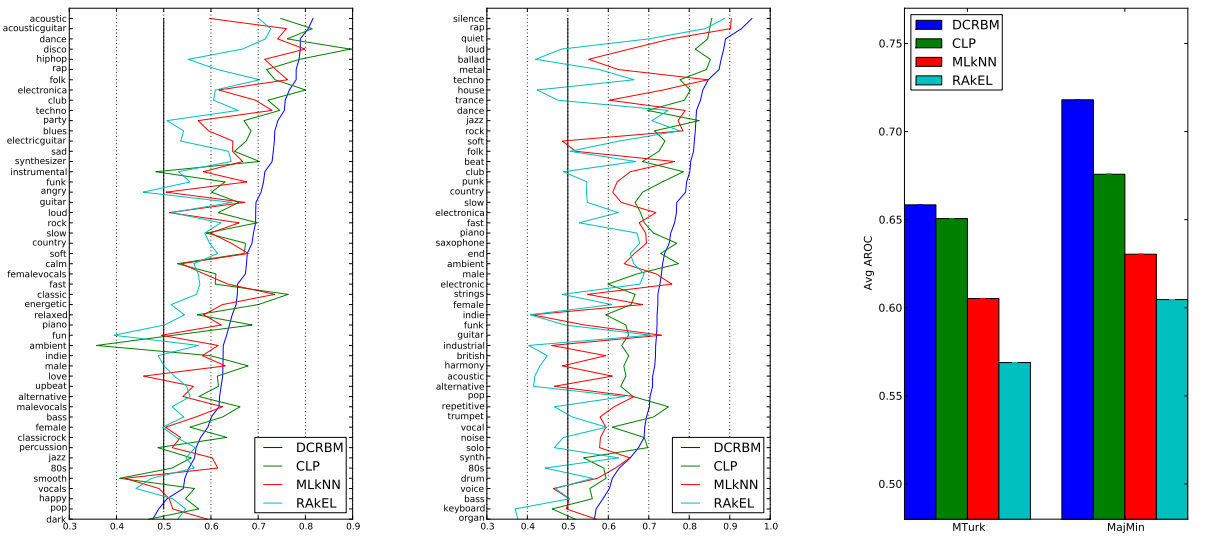
having the most tags applied to each clip, these tags are only applied at the track level, not at the clip level. Thus the assumption that tags apply to the clip in the middle of each track adds a certain amount of noise to the tags, as discussed in Section 2, and decreases the reliability of these tags as training data.

Figure 4 shows the per-tag breakdown of the performance of each algorithm on the 50 most popular tags from each dataset. The tags on the y-axis are sorted by the performance of the SVM using the doubly-conditional RBM smoothing, the best-performing autotagger.

On the Mechanical Turk dataset, shown in Figure 4(a), the DCRBM performs better than the other algorithms on almost all tags, and the performance of the other systems appears to be relatively correlated. Tags for which it performs much better than the other systems include **party**, **funk**, and **ambient**. The one tag on which it performs much worse than the other systems is **disco**. It is difficult to discern a pattern in these tags, but it seems like tags that are parts of clusters of related tags, i.e. tags with other contextual relatives, are classified better by the DCRBM.

On the MajorMiner dataset, shown in Figure 4(b), the DCRBM outperforms the other systems on most tags, and is not outperformed on any. Tags on which it performs particularly well are **quiet**, **metal**, **country**, and **alternative**. On this dataset it seems more clear that it does well on genre tags and other tags that have many contextual relatives, e.g. **quiet**.

On the Last.fm dataset, shown in Figure 4(c), the systems generally perform comparably. The DCRBM does much better on a few tags, namely **sad**, **chill**, **melancholic**, and **american**. It performs worse than the other systems on other tags, namely **downtempo**, **heavymetal**, and **postpunk**. In this case, the tags that it does well on seem to be related to **sad**, of which there are many. The tags it does worse on appear to be sub-genres that have fewer contextually-related tags and less co-occurrence with other tags.



(a) Mechanical Turk (b) MajorMiner (c) Overall

Fig. 5. Accuracy of autotaggers compared to baseline multi-label classifiers MLkNN and RAkEL. (c) Overall average AUC (after flipping AUCs below 0.5 to be above). AUC for each of the top 50 tags in the (a) Mechanical Turk and (b) MajorMiner datasets.

4.6 Experiment 4: Baseline systems

The final experiment compares the tag smoothing provided by our algorithms to three standard multi-label classification algorithms, random k-labelsets (RAkEL), multi-label k-nearest neighbors (MLkNN), and correlated label propagation (CLP). It uses the same data and procedures as experiment 3 above. We used the implementations of RAkEL and MLkNN from the Mulan library for multi-label learning [Tsoumakas et al. 2011] and implemented our own version of CLP.

RAkEL [Tsoumakas and Vlahavas 2007] is a meta-learning algorithm that transforms a multi-label classification problem into a set of multi-label problems each containing fewer labels. These smaller problems can then be treated as multi-class classification problems, treating the 2^N different label combinations as separate classes while avoiding the data sparsity of such an approach as N grows. We used the default parameter of subsets of 3 labels, that were not necessarily disjoint from one another. Because it is a meta-learner, we used it in conjunction with the default label powerset learner using the C4.5 classifier.

MLkNN [Zhang and Zhou 2007] is an extension of k-nearest neighbor classification to multi-label tasks. It estimates the probability of each label separately from the prevalence of each label among the nearest training points to a test point. We used the default number of nearest neighbors, 10. Correlated label propagation [Kang et al. 2006] is another extension of k-nearest neighbors to multi-label tasks. It explicitly considers the higher order correlations between labels, however, by carefully designing a cost function that can be optimized with a single pass through the labels, avoiding the exponential explosion of the label powerset. We found that on a validation dataset, ordering the weights of each class by increasing frequency and using the "exponential" kernel function performed best.

The results of these experiments can be seen in Figure 5. Figure 5(c) shows the average of the AUC results for each tag. Note that for the purposes of averaging AUC scores of the baseline algorithms, we took the complement of any AUC under 0.5 for a tag. The average AUC is still significantly below that achieved by the combination of doubly-conditional RBM-based smoothing and SVM classification.

The AUC for each of the top 50 tags can be seen in Figure 5(a) and (b) for the Mechanical Turk and MajorMiner datasets, respectively. On the Mechanical Turk dataset, the DCRBM achieved a higher AUC for a large majority of the tags except for the notable exceptions of **disco**, **electronica**, **classical**, and **energetic**. On the MajorMiner dataset, the DCRBM performed better than the baseline algorithms on all tags except for small differences on **saxophone**, **ambient**, **repetitive**, and **electronic**. This evidence solidly supports the conclusion that DCRBM is able to out-perform other multi-label classifiers on these tasks.

5. CONCLUSIONS

These experiments show that the autotagging of music becomes more accurate when augmented with contextual information, both in terms of tag co-occurrence context and temporal context. Temporal context can be employed because of the higher correlations shown to exist between the tags applied to clips that are "closer" to one another temporally. Tag-tag context can be applied because of synonymy and other contextual relationships between tags. These results are an important step in developing music classification and categorization beyond a series of isolated tag problems and clip examples to a more holistic approach that considers music and tags in their rich contextual relationships.

In addition to these ideas about context, this paper has contributed two new realizations of contextual tag language models in the form of conditional restricted Boltzmann machines. These models achieve better AUC performance than other state-of-the-art multi-label classifiers on this problem. It has also shown the value for capturing tag-tag context of information theoretic models that have previously been shown to accurately predict friendship links in social networks. Both of these modeling strategies could be applied to images, video, or many other modalities in which social tags are useful, but frequently sparse.

In the future, we would like to extend this work by modeling audio and tags jointly, instead of requiring separate tag-smoothing and classifier training stages. One candidate for such a model is the discriminative RBM [Larochelle and Bengio 2008; Mandel et al. 2011], which is closely related to the conditional RBMs employed in this paper. We would also like to exploit unlabeled and weakly labeled data, for example large collections of unlabeled music and large collections of text about music. RBMs and particularly discriminative RBMs look quite promising for such tasks. Finally, we would like to expand our use of information theoretic models beyond the tag-tag similarity to take advantage of the further context and richness available in the full set of (user, item, tag) triples.

ACKNOWLEDGMENTS

We are grateful to Last.fm for making their data available. This work was partly supported by the project *Social Integration of Semantic Annotation Networks for Web Applications* funded by National Science Foundation award IIS-0811994.

REFERENCES

AUCOUTURIER, J., PACHET, F., ROY, P., AND BEURIV, A. 2007. Signal + context = better classification. In *Proc. ISMIR*. 425–430.

BERTIN-MAHIEUX, T., ECK, D., MAILLET, F., AND LAMERE, P. 2008. Autotagger: A model for predicting social tags from acoustic features on large music databases. *J. New Music Res. 37,* 2, 115—135.

BESAG, J. 1975. Statistical analysis of non-lattice data. *The Statistician 24,* 3, 179–195.

BOUTELL, M., LUO, J., SHEN, X., AND BROWN, C. 2004. Learning multi-label scene classification1. *Pattern Recognition 37,* 9, 1757–1771.

CHEN, L., XU, D., TSANG, I. W., AND LUO, J. 2010. Tag-based web photo retrieval improved by batch mode re-tagging. 3440–3446.

CORTES, C. AND MOHRI, M. 2004. Auc optimization vs. error rate minimization. In *NIPS 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Vol. 16. MIT Press, Cambridge, MA.

ECK, D., LAMERE, P., BERTIN-MAHIEUX, T., AND GREEN, S. 2008. Automatic generation of social tags for music recommendation. In *NIPS 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. MIT Press, Cambridge, MA, 385–392.

HAN, Y., WU, F., JIA, J., ZHUANG, Y., AND YU, B. 2010. Multi-task sparse discriminant analysis (mtsda) with overlapping categories. In *AAAI Conference on Artificial Intelligence*. 469–474.

HAND, D. J. 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Mach. Learn. 77*, 103–123.

HEITZ, G. AND KOLLER, D. 2008. Learning spatial context: Using stuff to find things. In *ECCV*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Lecture Notes in Computer Science Series, vol. 4. Springer Berlin / Heidelberg, Berlin, Heidelberg, 30–43.

HINTON, G. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation 14*, 1771–1800.

HOIEM, D., EFROS, A., AND HEBERT, M. 2008. Putting objects in perspective. *International Journal of Computer Vision 80,* 1, 3–15.

KANG, F., JIN, R., AND SUKTHANKAR, R. 2006. Correlated Label Propagation with Application to Multi-label Learning. In *Intl. Conf. on Comp. Vision and Pat. Rec.* 1719–1726.

LAROCHELLE, H. AND BENGIO, Y. 2008. Classification using discriminative restricted Boltzmann machines. In *Proc. ICML*, A. McCallum and S. Roweis, Eds. Omnipress, 536–543.

LEE, J. H. 2010. Crowdsourcing music similarity judgments using mechanical turk. In *Proc. ISMIR*. 183–188.

MANDEL, M., PASCANU, R., LAROCHELLE, H., AND BENGIO, Y. 2011. Autotagging music with conditional restricted boltzmann machines. Online: http://arxiv.org/abs/1103.2832.

MANDEL, M. I., ECK, D., AND BENGIO, Y. 2010. Learning tags that vary within a song. In *Proc. ISMIR*. 399–404.

MANDEL, M. I. AND ELLIS, D. P. W. 2008. A web-based game for collecting music metadata. *J. New Music Res. 37,* 2, 151–165.

MANNING, C., RAGHAVAN, P., AND SCHÜTZE, H. 2008. *Introduction to information retrieval*. Cambridge University Press.

MARKINES, B., CATTUTO, C., MENCZER, F., BENZ, D., HOTHO, A., AND STUMME, G. 2009. Evaluating similarity measures for emergent semantics of social tagging. In *Proceedings of the 18th international conference on World wide web*. ACM, 641–650.

MIOTTO, R., BARRINGTON, L., AND LANCKRIET, G. 2010. Improving auto-tagging by modeling semantic co-occurrences. In *Proc. ISMIR*. 297–302.

MURPHY, K., TORRALBA, A., AND FREEMAN, W. T. 2004. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *NIPS 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, Cambridge, MA.

RABINOVICH, A., VEDALDI, A., GALLEGUILLOS, C., WIEWIORA, E., AND BELONGIE, S. 2007. Objects in context. In *Intl. Conf. on Computer Vision*. IEEE, 1–8.

RASIWASIA, N. AND VASCONCELOS, N. 2009. Holistic context modeling using semantic co-occurrences. In *Intl. Conf. on Comp. Vision and Pat. Rec.* IEEE, Los Alamitos, CA, USA, 1889–1895.

SALAKHUTDINOV, R., MNIH, A., AND HINTON, G. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proc. ICML*. 791–798.

SCHIFANELLA, R., BARRAT, A., CATTUTO, C., MARKINES, B., AND MENCZER, F. 2010. Folks in folksonomies: Social link prediction from shared metadata. In *Proc. ACM Intl. Conf. on Web search and data mining*. ACM, 271–280.

SLANEY, M. 2002. Semantic-audio retrieval. In *Proc. ICASSP*. Vol. 4.

SMOLENSKY, P. 1986. *Information processing in dynamical systems: foundations of harmony theory*. MIT Press.

SNOW, R., O'CONNOR, B., JURAFSKY, D., AND NG, A. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proc. Empirical Methods in NLP*. 254–263.

SOROKIN, A. AND FORSYTH, D. 2008. Utility data annotation with amazon mechanical turk. In *CVPR Workshops*. 1–8.

TAYLOR, G., HINTON, G. E., AND ROWEIS, S. 2007. Modeling human motion using binary latent variables. In *NIPS 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. MIT Press, Cambridge, MA, 1345–1352.

TINGLE, D., KIM, Y. E., AND TURNBULL, D. 2010. Exploring automatic music annotation with "acoustically-objective" tags. In *Proc. Intl. Conf. on Multimedia inf. retr.* ACM, 55–62.

TROHIDIS, K., TSOUMAKAS, G., KALLIRIS, G., AND VLAHAVAS, I. 2008. Multilabel classification of music into emotions. In *Proc. ISMIR*.

TSOUMAKAS, G., KATAKIS, I., AND VLAHAVAS, I. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Chapter 34, 667–685.

TSOUMAKAS, G., VILCEK, J., SPYROMITROS, L., AND VLAHAVAS, I. 2011. MULAN: a java library for multi-label learning. *Journal of Machine Learning Research*. Accepted for publication conditioned on minor revisions.

TSOUMAKAS, G. AND VLAHAVAS, I. 2007. Random k-Labelsets: An ensemble method for multilabel classification. In *Proc. ECML*. Lecture Notes in Computer Science Series, vol. 4701. Springer Berlin / Heidelberg, Berlin, Heidelberg, Chapter 38, 406–417.

WHITEHILL, J., RUVOLO, P., WU, T., BERGSMA, J., AND MOVELLAN, J. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS 22*, Y. Bengio, D. Schuurmans, C. Williams, J. Lafferty, and A. Culotta, Eds. 2035–2043.

WHITMAN, B. AND RIFKIN, R. 2002. Musical query-by-description as a multiclass learning problem. In *IEEE Workshop on Multimedia Signal Processing*. 153–156.

YAO, B. AND FEI-FEI, L. 2010. Modeling mutual context of object and human pose in human-object interaction activities. In *Intl. Conf. on Comp. Vision and Pat. Rec.* IEEE, 17–24.

ZHANG, M. AND ZHOU, Z. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition 40,* 7, 2038–2048.

Received September 2010; revised March 2011; accepted August 2011