# LEARNING TAGS THAT VARY WITHIN A SONG

**Michael I Mandel, Douglas Eck, Yoshua Bengio**
LISA Lab, Université de Montréal
{mandelm,eckdoug}@iro.umontreal.ca, yoshua.bengio@umontreal.ca

**ABSTRACT**

This paper examines the relationship between human generated tags describing different parts of the same song. These tags were collected using Amazon's Mechanical Turk service. We find that the agreement between different people's tags decreases as the distance between the parts of a song that they heard increases. To model these tags and these relationships, we describe a conditional restricted Boltzmann machine. Using this model to fill in tags that should probably be present given a context of other tags, we train automatic tag classifiers (autotaggers) that outperform those trained on the original data.

## 1. INTRODUCTION

Social tags are short free-form descriptions of music that users apply to songs, albums, and artists. They have proven to be a popular way for users to organize and discover music in large collections [5]. There remain, however, millions of tracks that have never been tagged by a user that cannot be included in these systems. Automatic tagging, based on an analysis of the audio of these tracks and user tagging behavior, could enable them to be included in these systems immediately. To this end, this paper explores the relationship between audio and the tags that humans apply to it, especially at different time scales and at different points within the same track.

We perform this examination in the context of a "Human Intelligence Task" (HIT) on the Mechanical Turk website [1], where users are paid small amounts of money to perform tasks for which human intelligence is required. Mechanical Turk has been used extensively in natural language processing [10] and vision [11, 13], but to our knowledge has not been used in music information retrieval before. Mechanical Turk is one means to the end of *human computation*, the field of cleverly harnessing human intelligence to solve computational problems. This field has been growing in popularity recently, especially in the context of games for collecting descriptions of music [6, 7, 12]. While these games have proven popular among researchers for collecting these data, they require significant investment of development time and effort in order to attract and retain players. By using Mechanical Turk, a researcher can trade a little extra money for significant savings in development time.

This paper makes three contributions. First, in Section 2 we discuss data collection and analysis from a new source, Mechanical Turk, and Section 2.1 shows that clips from different parts of the same song tend to be described differently from one another. Second, Section 3.1 presents a probabilistic model of tags and their relationships with each other to combat the sparsity of music tagging data. Section 3.3 shows that explicitly including information linking tags from the same user, track, and clip improves the likelihood of held out data under the model. Finally, we use this model to "smooth" tag data, i.e. to infer tags that were not provided, but perhaps should have been, given the tags that were. Section 4 shows that these smoothed tags are more "learnable" from the audio signal than the raw tags provided directly by the users, especially when fewer users have seen a given clip.

## 2. DATA COLLECTION

Users of the Mechanical Turk website, known as "turkers", were asked to listen to a 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, Other (sounds alike artists, era, locale, song section, audience, activities, etc.). In order to avoid biasing the turkers' responses, no examples of tags in each category 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 pre-

[1] http://mturk.com

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

| User | Track | Clip | Tags | Num pairs |
|---|---|---|---|---|
| + | + | + | $6.0370 \pm 0.0290$ | 2,566 |
| + | + | − | $2.3797 \pm 0.0511$ | 690 |
| + | − | − | $1.2006 \pm 0.0026$ | 227,006 |
| − | + | + | $1.1137 \pm 0.0142$ | 4,838 |
| − | + | − | $1.0022 \pm 0.0083$ | 13,560 |
| − | − | − | $0.5240 \pm 0.0004$ | 3,702,481 |

**Table 1**. Average number of tags ($\pm$ 1 standard error) shared by HITs with various characteristics in common and number of such pairs of HITs. A + indicates that the clips shared that characteristic, a − that they differed in it.

sented 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 pre-processed the data by transforming tags into a canonical form. We normalized the spelling of decades and the word "and", removed words like "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**.

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 Co-occurrence analysis

The first analysis that can be applied to these data is a simple counting of the number of tags shared by pairs of HITs. By categorizing the relationships between two HITs in terms of the users, tracks, and clips involved, an interesting picture emerges. Table 1 shows the first analysis of the number of shared tags for all possible pairs of HITs grouped by the relationships of these characteristics.

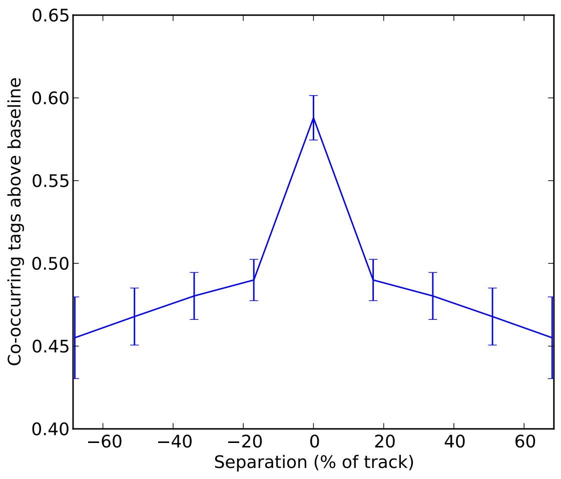


**Figure 1**. Average number of tags above the baseline shared by HITs from the same track as a function of the separation between the clips measured as % of a track.

The bottom row of the table shows that HITs with nothing in common still share 0.5240 tags on average because of the distribution of tags and music in this dataset. The second line from the bottom shows that HITs involving different users and different clips within the same track share 1.002 tags on average. And the third to last row shows that HITs with different users, but the same clip share 1.11 tags on average, significantly more than HITs that only share the same track. This same pattern also holds for HITs from the same user, but with higher co-occurrences. The large difference between HITs from the same user and HITs from different users can probably be attributed to the lack of feedback to the users in the task, allowing somewhat idiosyncratic vocabularies to perpetuate. Note that the top row of the table shows the average number of tags per HIT.

A related analysis can be performed measuring the dependence of tag co-occurrence on the distance between clips in the same track. Figure 1 shows the average tag co-occurrence of two clips in the same track above the baseline level of co-occurrences for two clips from different tracks. It reveals that the number of tags shared by clips decreases as the clips get farther apart. The error bars show that this result is not quite statistically significant, but it is still a notable trend. Results are similar for HITs from the same user and for cosine similarity instead of plain co-occurrence.

## 3. DATA MODELING

While stemming can make connections between certain tags in the dataset, it is only able to do this for tags which are syntactically related to one another. Another kind of model is required to capture relationships between tags like **indie** and **rock**. We choose to capture these relationships using a restricted Boltzmann machine (RBM), a generative probabilistic model. The RBM observes binary vectors representing the tags that a single user gave to a single clip. Once trained, the model can compare the relative likelihood of two such observations and can draw samples from the observation distribution.

## 3.1 Conditional restricted Boltzmann machine

More formally, an RBM [9] is a probabilistic model of the relationship between binary visible units, denoted, $v_i$ and binary hidden units, denoted $h_j$. Conditioned on the visible units, the hidden units are independent of one another, and vice-versa. The joint probability density function is

$$p(v,h) = \frac{1}{Z} \exp\left(v^T W h + b^T v + c^T h\right) \tag{1}$$

where the partition function $Z \equiv \sum_{v,h} p(v,h)$ is computationally intractable (exponential either in the number of visibles or of hiddens). The likelihood of the observation $v$ is obtained by marginalizing over $h$: $p(v) = \sum_h p(v,h)$, and can be computed easily up to $Z$. In this paper, we condition the model on "auxiliary" hidden units, $a$,

$$p(v,h \mid a) = \frac{1}{Z} \exp\left(v^T W h + v^T W_a a + b^T v + c^T h\right) \tag{2}$$

where the partition function is now conditioned on $a$ as well, $Z = \sum_{v,h} p(v,h \mid a)$. Conditional RBMs have been used for collaborative filtering [8], although in that case the conditioning variables influenced the hidden states, whereas in our model they directly influence the visible units. The matrices $W$ and $W_a$ and the bias vectors $c$ and $b$ are learned using the contrastive divergence algorithm [4]. In addition to the normal contrastive divergence updates, we place an $L_1$ penalty on $W_a$ to promote sparseness of its entries.

In practice, the vector $a$ is set *a priori* to represent the user, the artist, the track, and/or the clip using a so-called *one hot* representation. For example, each user has their own column of the $W_a$ matrix, providing a different bias to the tag probabilities. We sometimes refer to the quantity $W_a a$ as the auxiliary biases for this reason. Each user in effect has a different baseline probability for the visible units, meaning that they tend to use the tags in different proportions. Because the entries of the $W_a$ matrix are $L_1$-penalized, the user columns tend to represent discrepancies between a user's tags and the global average, which is captured in the bias vector $b$. Thus the $W_a$ matrix is like a term frequency-inverse document frequency (TF-IDF) representation (see e.g. [14]) of the variables that it is modeling, but learned in a more probabilistically grounded way.

## 3.2 Purely textual datasets

We apply this model to three different tag datasets with the goal of discovering relationships between tags, and the tags that are used unexpectedly frequently or infrequently on particular items. The first dataset is purely textual, from Last.fm [1]. It includes (artist, tag) pairs, along with the number of times that that pair appears. The second dataset, from MajorMiner [7], includes (clip, user, tag) triples and also includes the audio associated with each clip. The third dataset, from the Mechanical Turk experiments described in Section 2, similarly includes (clip, user, tag) triples and audio. While it is smaller than the MajorMiner data, it includes many more clips per track, and so can provide perhaps more insight into clip-level and track-level modeling.

The dataset from [1] was collected from Last.fm in the spring of 2007. It includes the tags that users applied to approximately 21,000 unique artists and the number of users who applied each tag to each artist. There are approximately 100,000 unique tags, and 7.2 million (artist, tag) pairs, including duplicates. To reduce the size of the required model, we discarded tags that had been applied to fewer than 8000 artists (98 tags), and only kept the 200 most frequently tagged artists.

In order to transform this dataset into a form that can be used by the RBM model, we simulated taggings from individual users. We characterized each artist with independent Bernoulli probabilities over each tag and drew multi-tag samples from this distribution. The probability of each tag was proportional to the number of times each tag was applied to an artist, so the counts were first normalized to sum to 1. These normalized counts were multiplied by 5 (and truncated to prevent probabilities greater than 1) so that the expected total number of tags was 5, a number that a typical user might provide. To create the dataset, we repeatedly drew an artist at random and simulated a user's tagging of that artist. The artists' tag probabilities provided a baseline against which to measure the estimation of the relevant $W_a$ columns, which only modeled artist auxiliary information.

The dataset from [7] was collected from the MajorMiner music labeling game over the course of the last three years. It includes approximately 80,000 (clip, user, tag) triples with 2600 unique clips, 650 unique users, and 1000 unique tags. Each observation was encoded as a binary vector indicating the tags that a single user applied to a single clip. The $a$ vector in this case indicated both the clip, the track that it came from, and the user. On average, each track was represented by fewer than two clips.

Finally, this new Mechanical Turk dataset provides (clip, user, tag) triples along with relationships between clips and tracks. While it contains the fewest triples, it contains the most structure of the datasets because by design there are five clips per track. To model it, the $a$ vector represents the user, the track, and the clip, so there is a separate auxiliary term learned for each of them.

## 3.3 Textual experiments

Qualitative experiments on the Last.fm dataset showed that our model successfully learned the auxiliary inputs, i.e. the $W_a$ matrix acted as a sort of TF-IDF model for tags. Specifically, the $W$ matrix modeled relationships between pairs of tags, the $b$ vector modeled overall popularity of individual tags, and the columns of $W_a$ modeled any tags that were unusually prevalent or absent for an artist given its other tags. For example, Nirvana's $W_a$ column included a large value for **grunge**, and the Red Hot Chili Peppers' included a large value for **funk**, both of which might not have been expected from their other tags like **rock** and **alternative**. Similarly, the Beatles have a negative bias for **seen live** because presumably fewer Last.fm listeners have seen the Beatles live than other artists tagged **rock** and **pop**. These issues are addressed more quantitatively below.

All three of the datasets described in Section 3.2 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 [2]. 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

$$\mathrm{PL}(v \mid a) \equiv \prod_i p(v_i \mid v_{\setminus i}, a) = \prod_i \frac{p(v \mid a)}{p(v \mid a) + p(\tilde{v}_i \mid a)} \tag{3}$$

where $v_i$ is the $i$th visible unit, $v_{\setminus i}$ is all of the visible units except for the $i$th unit, and $\tilde{v}_i$ is the observation $v$ with the $i$th 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, than 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 models 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. 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

| | Auxiliary info | | | |
|---|---|---|---|---|
| Dataset | User | Track | Item | log(PL) ± 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 |
| Last.fm | − | − | + | −0.5623±0.0042 |
| Last.fm | − | − | − | −0.7082±0.0029 |
| Last.fm | | baseline | | −1.1825±0.0018 |

**Table 2**. 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.

these systems are shown in Table 2. 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 also models the relationships between tags through its hidden units. 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. AUTOTAGGING EXPERIMENTS

The final set of experiments involves not just the textual tags, but also the audio for both the MajorMiner dataset and this new data collected from Mechanical Turk. In this experiment, we measure the usefulness of the RBM model from Section 3.1 for "smoothing" the tag data. Specifically, we create two datasets: the first, labeled "raw", consists of just the original (clip, user, tag) triples in the dataset. The second, labeled "smoothed", consists of labels imputed by the RBM trained with all of the available auxiliary information. For each clip, we drew 1000 samples from the RBM conditioned on that sample's auxiliary information, but with no user indicated. We factored out the user so the taggers were trained from a general point of view, not that of any

Mechanical Turk

| | Tested | |
|---|---|---|
| Trained | Raw | Smoothed |
| Raw | 56.87 ± 0.52 | 56.56 ± 0.36 |
| Smoothed | 61.43 ± 0.51 | 63.40 ± 0.35 |

MajorMiner

| | Tested | |
|---|---|---|
| Trained | Raw | Smoothed |
| Raw | 65.97 ± 0.49 | 60.58 ± 0.35 |
| Smoothed | 66.67 ± 0.49 | 63.09 ± 0.35 |

**Table 3**. Average classification accuracy and standard errors of autotaggers trained and tested on different tag labelings for Mechanical Turk and MajorMiner data. The tags were either raw or smoothed from RBM samples.

particular user. Because the model assumes the effects of user, track, and clip are additive on the tag probabilities, the effect of one can be factored out by not adding it. This is further ensured by the regularization of the $W_a$ matrix, which forces many of the elements of the matrix to 0 and the rest to be small.

To compare these datasets, we hold the acoustic features constant, but change the labels used to train and test classifiers. We first split the data into 5 cross-validation folds. Then the positive and negative test examples for a particular tag are the top- and bottom-ranked clips from one cross-validation fold. The training examples are the top- and bottom-ranked clips excluding that fold. Because the cross-validation breakdowns are preserved across tag sets, it is possible to train on one tag set and test on another. For the smoothed dataset, we select the top and bottom 100 examples for each tag. For the raw counts, we choose for each tag the smaller of the top 100 examples or all of the examples verified by at least 2 people.

The autotaggers are inspired by those from [7], which use timbral and rhythmic features and a support vector machine (SVM) classifier. For this experiment we use LibSVM's $\nu$-SVM as our SVM implementation, with probability estimates and a linear kernel [3]. Performance with the Gaussian kernel was similar. One binary SVM is trained per tag using a balanced number of positive and negative examples selected in order of tag affinity in the training set. Performance is measured in terms of average accuracy on a test dataset that is balanced in terms of positive and negative examples to set a constant baseline of 50% for a randomly guessing classifier. This metric is more appropriate than overall classification accuracy for tasks like autotagging where it is important to recognize positive examples in the presence of a large number of negative examples. To avoid the "album effect", the cross-validation folds were assigned so that clips from the same track were in the same fold in the Mechanical Turk data and that clips from the same album were in the same fold in the MajorMiner data.

The results of these experiments are shown in Table 3 and Figure 2. Each row of the tables represents a training tag labeling and each column represents a test tag labeling. The tables show these accuracies averaged over the 95 tags used by the most people on each dataset. The first column of each table shows the result of training on different tag labelings and testing on the raw tags. For both the MajorMiner and Mechanical Turk datasets, smoothing with the RBM improves test performance on the raw, user-supplied tags, although for the MajorMiner dataset, this difference is not statistically significant. The second column of each table indicates the performance of both models in predicting the smoothed data. In this case as well, the smoothed data trains more accurate models.

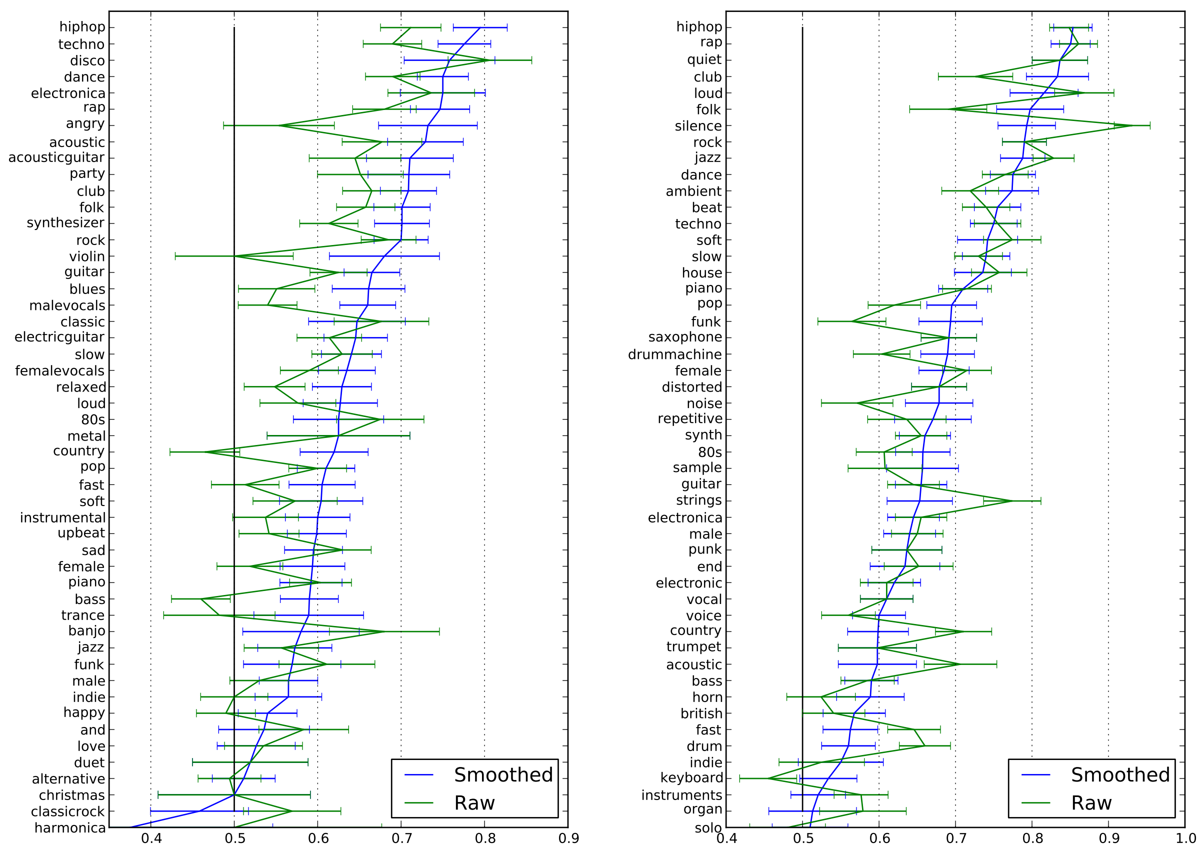
The diagonals of these tables show the "learnability" of the tag labelings. For the Mechanical Turk dataset, the smoothed tag set is more learnable than the raw tags. For the MajorMiner dataset, however, the raw tags are more learnable than the smoothed tags. These accuracies may not be directly comparable, however, because the measurements differ in both the models used and the test data. The difference in accuracy might indicate that the smoothing is less necessary in the MajorMiner dataset due to its larger size and larger number of repeated (clip, tag) pairs.

Figure 2 shows the autotag classification accuracy on the raw tags when trained with the raw and smoothed tags. The tags shown are the 50 used by the most people, and are sorted in the plots by the performance of the best system, that trained on the smoothed tags. For the Mechanical Turk data, shown in Figure 2(a), these smoothed tags train better classifiers almost across the board. Certain tags perform slightly better when trained on the raw data, but not significantly so. Smoothing is particularly useful for training **angry**, **violin**, and **country**, where autotaggers trained from the raw tags perform at chance levels.

For the MajorMiner data, shown in Figure 2(b), the smoothed tags and the raw tags perform similarly to one another. The smoothed tags train better autotaggers for **club**, **folk**, **pop**, and **funk**, while the raw tags train better autotaggers for **silence**, **strings**, **country**, and **acoustic**. The occurrence of the **silence** tag was due to the inclusion of a few broken clips in the game, which makes it a very specific, context-dependent tag that the RBM might not be able to generalize. It is not clear why performance on **country** is so different between the two datasets. It could be because in the Mechanical Turk dataset the top co-occurring tags with **country** are **guitar** 61% of the time and **folk** 27%, while in MajorMiner, they are **guitar** 44% of the time, **female** 27%, and **male** 26%. Thus in Mechanical Turk smoothing gives better results for **country** because it occurs more frequently with **guitar** and occurs with the more informative tag **folk**.

## 5. CONCLUSION

This paper has discussed the relationships between tags and music at a sub-track scale. We found that Mechanical Turk was a viable means of collecting ground truth tag data from humans, although the lack of the immediate feedback of a game might have contributed to lower inter-user agreement.



(a) Mechanical Turk

(b) MajorMiner

**Figure 2**. Accuracy of autotaggers for the top 50 tags in the Mechanical Turk and MajorMiner datasets. The autotaggers were trained on raw and smoothed tags and tested on the raw, human generated tags. Error bars show 1 standard error.

We also found that different parts of the same song tend to be described differently, especially as they get farther from one another. By modeling these differences with a conditional restricted Boltzmann machine, we were able to recover false negative tags in the user-generated data and use these data to more effectively train autotaggers, especially in smaller datasets. In the future we will explore additional models of tag-tag similarity, joint tag-audio models, and models of tagging that take into account the relationships between clips' different distances from one another.

**Acknowledgements** The authors acknowledge the support of an NSERC Discovery grant and would like to thank Razvan Pascanu and Johanna Devaney for their assistance.

## 6. REFERENCES

[1] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: A model for predicting social tags from acoustic features on large music databases. *J. New Music Res.*, 37(2):115–135, 2008.

[2] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

[3] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

[5] P. Lamere. Social tagging and music information retrieval. *J. New Music Res.*, 37(2):101–114, 2008.

[6] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie. Evaluation of algorithms using games: the case of music annotation. In *Proc. ISMIR*, pages 387–392, 2009.

[7] M. I. Mandel and D. P. W. Ellis. A web-based game for collecting music metadata. *J. New Music Res.*, 37(2):151–165, 2008.

[8] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proc. ICML*, pages 791–798, 2007.

[9] P. Smolensky. *Information processing in dynamical systems: foundations of harmony theory*. MIT Press, 1986.

[10] 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 *Proc. Empirical Methods in NLP*, pages 254–263, 2008.

[11] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *CVPR Workshops*, pages 1–8, 2008.

[12] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. In *Proc. ISMIR*, pages 225–230, 2008.

[13] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS 22*, pages 2035–2043, 2009.

[14] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34, 1998.