AUTOTAGGING MUSIC USING SUPERVISED MACHINE LEARNING

Douglas Eck
Sun Labs
Sun Microsystems
Burlington, Mass, USA
douglas.eck@umontreal.ca

Thierry Bertin-Mahieux
Univ. of Montreal
Dept. of Comp. Sci.
Montreal, QC, Canada
bertinmt@iro.umontreal.ca

Paul Lamere
Sun Labs
Sun Microsystems
Burlington, Mass, USA
paul.lamere@sun.com

ABSTRACT

Social tags are an important component of "Web2.0" music recommendation websites. In this paper we propose a method for predicting social tags using audio features and supervised learning. These automatically-generated tags (or "autotags") can furnish information about music that is untagged or poorly tagged. The tags can also serve to smooth the tag space from which similarities and recommendations are made by providing a set of comparable baseline tags for all tracks in a recommender system.

1 INTRODUCTION

Social tags are a key part of "Web 2.0" technologies and have become an important source of information for recommendation. In the domain of music, websites such as Last.fm [1] use social tags as a basis for recommending music to listeners. In this paper we propose a method for predicting social tags using audio feature extraction and supervised learning. These automatically-generated tags (or "autotags") can furnish information about music for which good, descriptive social tags are lacking. The tags can also serve to smooth the tag space from which similarities and recommendations are made by providing a set of comparable baseline tags for all tracks in a recommender system.

In this paper we investigate the automatic generation of tags with properties similar to those generated by social taggers. Specifically we introduce a machine learning algorithm that takes as input acoustic features and predicts social tags mined from the web (in our case, Last.fm). The model can then be used to tag new or otherwise untagged music, thus providing a (partial) solution to the cold-start problem. We believe these autotags might also serve to dampen feedback loops which occur when certain songs in a social recommender become over-popular and thus over-tagged. This paper represents preliminary work: we have tested our approach on a sufficient number of tagsets to conclude that the approach has merit. However we have yet to test the output of the model in a social tagging system. Thus we are unable to say at this time how well such an approach will perform with embedding new songs or with "smoothing" recommendations in the event of feedback loops.

Our paper is organized as follows: in Section 2 we discuss social tags in more depth, including a discussion of the tag set we built for these experiments. In Section 3 we present an algorithm for autotagging songs based on labeled data acquired on the Internet using data mining techniques. In section Section 4 we present experimental results and also discuss the ability to use model results for visualization.

2 SOCIAL TAGGING

Recently, there has been increasing interest in *social tagging* [8] including the social tagging of music. Music tagging sites such as Last.fm and QLoud [2] allow music listeners to apply free-text labels (tags) to songs, albums or artists. Typically, users are motivated to tag as a way to organize their own personal music collection. A user may tag a number of songs as "mellow" some songs as "energetic" some songs as "guitar" and some songs as "punk" The labels may overlap (a single song may be labeled "energetic" "guitar" and "punk" for example). Typically, a music listener will use tags to help organize their listening. A listener may play their "mellow" songs during the evening meal, and their "energetic" artists while they exercise.

The real strength of a tagging system is seen when the tags of many users are aggregated. When the tags created by thousands of different listeners are combined, a rich and complex view of the song or artist emerges. Table 1 show the top 20 tags and frequencies of tags applied to the band "The Shins" Users have applied tags associated with the genre (Indie, Pop, Rock, Emo, Folk etc.), with the mood (mellow, chill), opinion (favorite), style (singer-songwriter) and context (Garden State). From these tags and their frequencies we learn much more about "The Shins" than we would from a traditional single genre assignment of "Indie Rock". Additionally, in previous work [3] it was shown that social tags (in this case from the freedb CD track listing service at `www.freedb.org`) can predict canonical music-industry genre with good accuracy. Thus we lose little and gain a lot by moving from

[1] http://last.fm

[2] http://www.qloud.com

genres to tags.

For this research, we extracted tags and tag frequencies for over 50,000 artists from the social music website Last.fm using the Audioscrobbler web service [1]. Table 2 shows the distribution of the types of tags for the 500 most frequently applied tags. The majority of tags describe audio content. Genre, mood and instrumentation account for 77% of the tags. This bodes well for using the tags to predict audio similarity as well as using audio to predict social tags. However, there are numerous issues that can make working with tags difficult. Taggers are inconsistent in selecting tags, using synonyms such as "favorite", "favourite" and "favorites", "hip hop" "hiphop" and "hip-hop". Taggers use personal tags that have little use when aggregated ("i own it", "seen live'). Tags can be ambiguous; "love" can mean a romantic song or it can mean that the tagger loves the song. Taggers can be malicious, purposely mistagging items (presumably there is some thrill hearing lounge singer Barry Manilow included in a death metal playlist). Taggers can purposely mistag items in an attempt to increase or decrease the popularity of an item. Although these issues make working with tags difficult, they are not impossible to overcome. Some strategies to deal with these are described in [6].

When social tags are used as a part of collaborative filtering systems, there is also the problem of social feedback loops: a song can become popular simply because a few people start recommending it. This was documented in [10], where a number of artificial music markets were created. Increasing social influence in the system resulted in unequal and unpredictable performance. In short, popular songs become more popular while unpopular songs become more unpopular. Also, in different social influence worlds (as created in the experiment), a particular song might move to #1 or languish at the bottom of the charts. Perhaps related, in the social recommender Last.fm, a song by The Postal Service remained at number one on the charts for 6 months, seemingly due to a user-recommendation feedback loop[7].

A more difficult issue is the uneven coverage and sparseness of tags for unknown songs or artists. Since tags are applied by listeners, it is not surprising that popular artists are tagged much more frequently than non-popular artists. In the data we collected from Last.fm, "The Beatles" are tagged 30 times more often than "The Monkees". This sparseness is particularly problematic for new artists. A new artist has few listeners, and therefore, few tags. A music recommender that uses social tags to make recommendations will have difficulties recommending new music because of the tag sparseness. This cold-start problem is a significant issue to address if we are to use social tags to help recommend new music.

Overcoming the cold-start problem is the primary motivation for this area of research. For new music or sparsely tagged music, we predict social tags directly from the audio and apply these automatically generated tags (called *autotags*) in lieu of traditionally applied social tags. By automatically tagging new music in this fashion, we can reduce or eliminate much of the cold-start problem.

Tag	Freq	Tag	Freq
Indie	2375	Mellow	85
Indie rock	1138	Folk	85
Indie pop	841	Alternative rock	83
Alternative	653	Acoustic	54
Rock	512	Punk	49
Seen Live	298	Chill	45
Pop	231	Singer-songwriter	41
The Shins	190	Garden State	39
Favorites	138	Favorite	37
Emo	113	Electronic	36

Table 1. Top 20 tags applied to *The Shins*

Tag Type	Frequency	Examples
Genre	68%	heavy metal, punk
Locale	12%	French, Seattle, NYC
Mood	5%	chill, party
Opinion	4%	love, favorite
Instrumentation	4%	piano, female vocal
Style	3%	political, humor
Misc	3%	Coldplay, composers
Personal	1%	seen live, I own it

Table 2. Distribution of tag types

3 AN AUTOTAGGING ALGORITHM

We now describe a machine learning model which uses the *meta-learning* algorithm AdaBoost [4] to predict tags from acoustic features. This model is an extension of a previous model [2] which performed well at predicting music attributes from acoustic features: at MIREX 2005 (ISMIR conference, London, 2005) the model won the Genre Prediction Contest and was the 2nd place performer in the Artist Identification Contest. The model has two principal advantages. First it performs automatic feature selection based on a feature's ability to minimize empirical error. Thus we can use the model to eliminate useless feature sets. Second, it's performance is linear in the number of inputs. Thus it has the potential to scale well to large datasets. (To be clear: this performance is task dependent and has not been conclusively demonstrated. However it is a motivation for our use of AdaBoost). Both of these properties are general to AdaBoost and are not explored further in this short paper. See [4, 11] for more.

3.1 Acoustic feature extraction

We generated MP3s from a subset of the tagged artists described in Section 2. From these MP3s we extracted several popular acoustic features. Do to space limitations, we do not cover feature extraction in depth here. Please see [2] for details. The features were extracted with high

temporal precision to preserve spectral and timbral information. Following the strategy of [2] coarser "aggregate" features were generated by taking means and standard deviations of high-temporal precision features over longer timescales. For our work here the timescale used was 5sec, a value consistent with the results of our previous work. When fixing hyperparameters for these experiments, we also tried a combination of 5sec and 10sec features, but saw no real improvement in results.

The features used included 20 Mel-Frequency Cepstral Coefficients, 176 autocorrelation coefficients computed for lags spanning from 250msec to 2000msec at 10ms intervals, and 85 spectrogram coefficients sampled by constant-Q (or log-scaled) frequency. We also tried 12 chromagram coefficients but discarded them because (not surprisingly) they contributed very little to the final result. For those not familiar with these standard acoustic features, please see [5].

3.2 Labels as a classification problem

Intuitively, automatic labeling would be a regression task where a learner would try to predict tag frequencies for artists or songs. However, because tags are sparse (many artist are not tagged at all) this proves to be too difficult using our current Last.fm / Audioscrobbler dataset. Instead we chose to treat the task as a classification one. Specifically, for each tag we try to predict if a particular artist has "none", "some" or "a lot" of a particular tag relative to other tags.

Note that we use a relative measure. That is, our tags for a given artist are normalized so that artists having many tags can be compared to artists having few tags. Then for each tag, an artist was decided as being "none", "some" or "a lot" depending on the proportion of times a particular tag was assigned to that artist relative to other tags assigned to that artist. Thus if an artist received only 50 "rock" tags and nothing else, it would be treated as having "a lot" of rock. Conversely, if an artist received 5000 "rock" tags but 10,000 "jazz" tags it would be treated as having "some" "rock" and "a lot" of "jazz". The specific boundaries between "none", "some" and "a lot" were decided by summing the normalized tag counts or all artists, generating a 100-bin histogram for each tag and moving the category boundaries such that an equal number of artists fall into each of the categories. In the case of "classical", the tagging was so sparse that this approach yielded only two bins because there were so many "none" values for "classical" that it was not possible to divide the remaining tagged "classical" examples into two bins of significant size. In Figure 1 the histogram for "rock" is shown (with only 30 bins to make the plot easier to read). Note that most artists are not labeled rock (bin 0) and that otherwise most of the mass is in high-bins. This was the trend for most tags and one of our motivations for using a 3-bin approach.

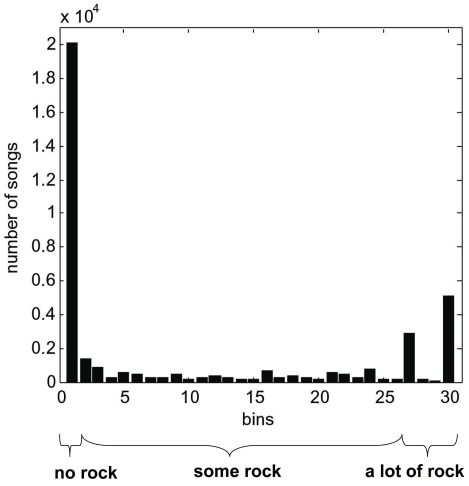

Figure 1. A 30-bin histogram of the proportion of "rock" tags against all other tags.

3.3 Tag prediction with AdaBoost

We now describe how to train an AdaBoost-based classifier to predict the tags based on labeled data. For those interested in more details about how the learning works please see [2], which also used AdaBoost.MH and aggregate features.

In short, audio features are extracted from a song. Aggregate features spanning 5 seconds of music are generated. If a song is longer than 5 minutes, only the middle 5 minutes of data are kept (for reasons of computational efficiency). Using MultiBoost.MH a booster is trained to predict the tag ("none", "some", "a lot") directly from the aggregate feature values. The value for a song is taken by voting over the predictions for each aggregate feature. Voting can take place in two ways: we can choose segment winners and then select as global winner the class receiving the most segment votes or we can sum the weak learner values over segments and then take the class with the maximum sum.

3.4 AdaBoost and AdaBoost.MH

AdaBoost [4] is a *meta-learning* method that constructs a *strong classifier* from a set of simpler classifiers, called *weak learners* in an iterative way. Originally intended for binary classification, there exist several ways to extend it to multiclass classification. We use AdaBoost.MH [11] which treats multiclass classification as a set of one-versus-all binary classification problems. In each iteration t, the algorithm selects the best classifier, called $h^{(t)}$ from a pool of *weak learners*, based on its performance on the training set, and assigns it a coefficient $\alpha^{(t)}$. The input of the *weak learner* is a d-dimensional observation vector $x \in \Re^d$ containing audio features for one segment of aggregated data (5 seconds in our experiments). The output of $h^{(t)}$ is a binary vector $y \in \{-1, 1\}^k$ over the k classes. $h_l^{(t)} = 1$ means a vote for class l by a *weak learner* while

$h^{(t)}$, -1 is a vote against. After T iterations, the algorithm output is a vector-valued discriminant function:

$$g(x) = \sum_{t=1}^{T} \alpha^{(t)} h^{(y)}(x) \quad (1)$$

Usually we obtain a single label by taking the class with the "most votes" i.e $f(x) = \arg\max_l g_l(x)$, but in our model, we use the output value for each class rather than the argmax.

Although we used our own implementation for these experiments, readers may use Multiboost [9] by Norman Casagrande to reproduce our results.

3.5 Generating autotags

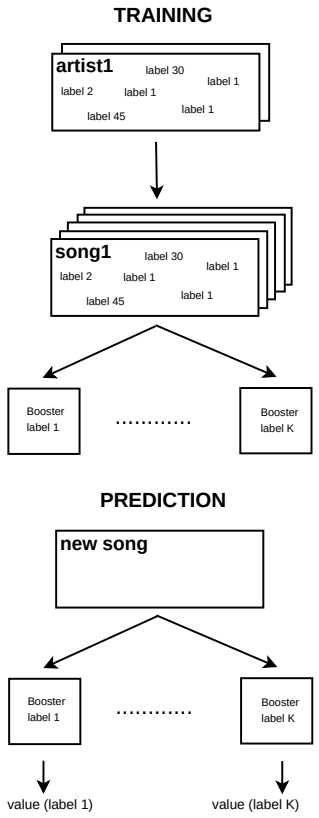

Figure 2. Overview of our model

Each booster yields a prediction over the bins (0="none", 1 = "some", 2 = "a lot") for a particular tag (e.g. "rock"). The predictions for individual tags can be used to generate autotags. Predictions for multiple tags can be used for visualization. To better understand the combination process, consider a boosted classifier built to predict "rock" tags. A typical output would be (0:-3.56) (1:0.14) (2:3.6) in format (class:value). A naive way to obtain only one value for "rock" is to take the barycenter (center of mass) of these values. However because the values are not scaled well with respect to one another, we ended up with poorly scaled results. Another intuitive idea is simply to subtract the value of the "none" bin from the value of the "a lot" bin, the reasoning being that "none" is truly opposite of "a lot". In our example this would yield a "rock" strength of 7.16. Thus to generate our final measure of "rock"-ness, we ignore the middle bin ("some"). However this should not be taken to mean that the middle "some" bin is useless: the booster needed to learn to predict "some" during training thus forcing it to be more selective in predicting "none" and "a lot". As a large-margin classifier, AdaBoost tries to separate the classes as much as possible. Thus the magnitude of the values for each bin are not easily comparable. To remedy this we normalize by taking the min and max prediction for each booster, which seems to work for finding similar artists. However, this does not seem to help with visualization where slightly better results are had using no normalization (when MDS is employed). This normalization would not be necessary if we had good tagging data for all artists and could perform regression on the frequency of tag occurrence across artists.

4 EXPERIMENTS

To test our model we used a subset of data from Last.fm / Audioscrobbler. From the full set of tags we selected 13 tags corresponding to popular genres. We selected these particular tags to be relatively easy to analyze (i.e. it's not clear how to analyze the performance of a predictor of "fun" or "mellow"). The tags we selected are shown in Table 3.

13 selected tags from Last.fm			
Tag	# Artists	Tag	# Artists
jazz	225	soul	44
rock	185	alternative	41
electronic	115	country	15
classical	108	punk	15
folk	59	reggae	14
indie	55	britpop	10
classic rock	51		

Number of artists with at least one tag	937
Number of artists with no tag	605
Number of songs	43597

Table 3. Statistics on the tags we data-mined from Last.fm. "# Artists" indicates the number of artists for which this tag was the most frequent.

4.1 Prediction Results

As described above, a classifier was trained for each of the 13 tags. Classification errors are reported in Table 4. For comparison we computed a baseline using the model from [2]. Unlike our current approach, this model assumes that one and only one tag can be applied to a single song. In order to train and test the model we needed

to select a winner. We simply chose the most frequent tag. The classification error rate was relatively high but significantly better than chance for 13-class classification: 62% error by segment and 59% error by song. The error rates for our boosted classifiers are shown in Table 4. As can be seen the performance is significantly better than the single-label model.

Summary of error rates		
	Segments	Songs
alternative	44.64	40.72
britpop	40.87	37.74
classic rock	43.31	39.57
classical	13.20	10.0
country	41.14	35.15
electronic	42.05	38.88
folk	40.11	37.36
indie	46.29	42.48
jazz	42.01	37.53
punk	40.63	36.65
reggae	38.73	35.18
rock	45.15	41.28
soul	44.5	41.16

Table 4. Test error (%) on predicting bins for songs and segments. Note that classical has only 2 bins. All boosters used 2000 single-stump weak learners.

In Table 5 we compare the top artists for "classical", and "rock" tags for both Last.fm tags and predicted tags. Unsurprisingly, the top artists for the Last.fm tags are popular artists. This popularity bias is less evident in the top artists for the predicted tags. This highlights the possibilities of using predicted tags to reduce the popularity bias inherent in social recommenders.

In Table 6 we compare the nearest neighbors for our predicted tags to those for the original tags. Unfortunately there is not enough space to provide more examples. In general it seems that our predicted tags are comparable in quality to the original tags. That is, our tags have some surprising errors (Marvn Gaye as a near neighbor to the Beatles?) yet so do the original tags (John Williams as a near neighbor to Mozart?).

4.2 Visualization

In our experiments we used 13 labels and boosters, so each artist has a dimension 13. We tried reducing these dimensions down to 2 or 3 using multi-dimensional scaling (MDS). We also tried the same on the Last.fm tags. The results, not shown here, are not easy to interpret. One always finds a few close artists which are good matches. At the same time there were many examples where the 2D embedding simply did not work. More promising results were had by visualizing how different bands fit into the 2D space made by combining two boosters. In Figure 3 see how popular artists fit into the space of "rock" versus "electronic". For readers familiar with the artists, it is easy to see that the results are, if not perfect, reasonable.

Last.fm tags	Predicted tags
classical	
Ludwig van Beethoven	William Byrd
Johann Sebastian Bach	Gioacchino Rossini
Frederic Chopin	Sir Edward Elgar
Antonio Vivaldi	Thomas Talis
Johannes Brahms	Robert Schumann
Claude Debussy	Claudio Monteverdi
Franz Schubert	Wolfgang Amadeus Mozart
Edvard Grieg	Ralph Vaughn Williams
rock	
Foo Fighters	Swervedriver
Radiohead	Gin Blossoms
Nirvana	peach
The Beatles	Echobelly
Led Zeppelin	Built to Spill
System of a Down	Ministry
Pink Floyd	Alien Ant Farm
Weezer	Stereolab

Table 5. Top artists for two tags, given by Last.fm tags and predicted tags. No normalization was used.

Near-neighbor artists		
Seed Artist	Last.fm Tags	Our Prediction
Fatboy Slim	The Prodigy Basement Jaxx Apollo 440	Chemical Brothers Apollo 440 Beck
The Beatles	John Lennon The Beach Boys The Doors	Eric Clapton Marvin Gaye The Rolling Stones
Mozart	Bach Beethoven John Williams	Schubert Haydn Brahms

Table 6. A comparison of 3 nearest neighbors for Last.fm tags versus our model predictions. Euclidean distance was used.

5 CONCLUSION AND FUTURE WORK

The work presented here is preliminary. However we conclude that a supervised learning approach to autotagging has merit. Our predictions are noisy and lead to sometimes-counterintuitive low-dimensional visualizations. However social tags themselves share these properties. There is much future work to do. One next step is to compare the performance of our boosted model to other approaches such as SVMs and neural networks. The dataset used for these experiments is already larger than published results for genre and artist classification. However, another order of magnitude of data are necessary to approximate even a small commercial database of music. Though the MDS

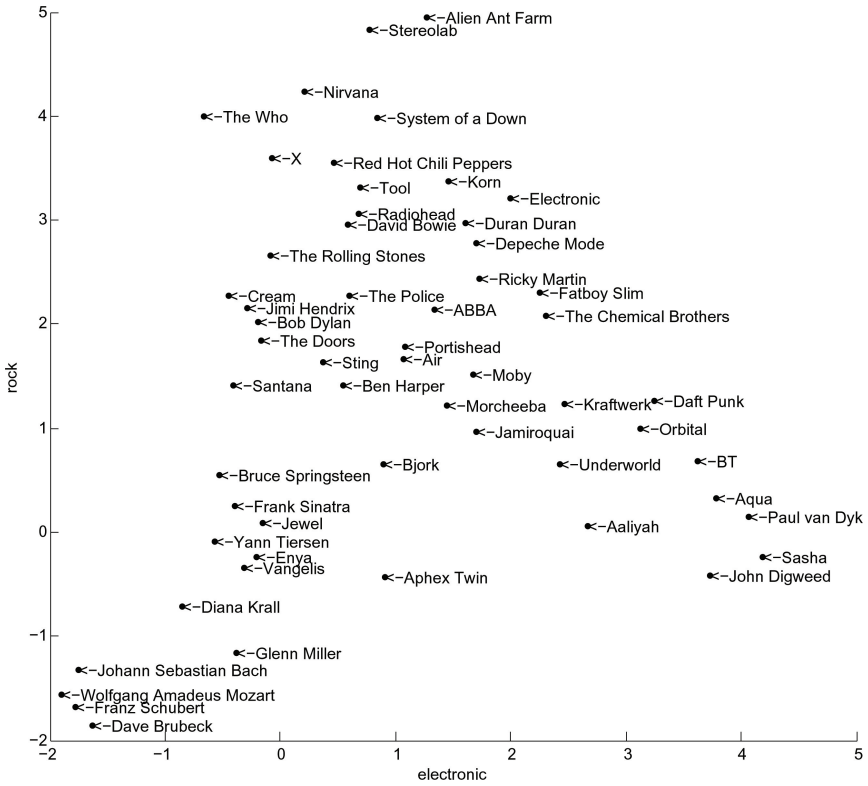

Figure 3. "rock" versus "electronic" for several artists.

visualization results are difficult to interpret, the visualizations of tag space (autotag and otherwise) are interesting. Better dimensionality reduction techniques for these data may yield useful music visualization tools based on autotags. Most importantly, the machine-generated autotags need to be tested in a social recommender. It is only in such a context that we can explore whether autotags, when blended with real social tags, will in fact yield improved recommendations.

Acknowledgments

Thanks to Balazs Kegl and James Bergstra for many discussions about boosting and genre classification.

6 REFERENCES

[1] Audioscrobbler. Web Services described at http://www.audioscrobbler.net/data/webservices/.

[2] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 65(2-3):473–484, 2006.

[3] J. Bergstra, A. Lacoste, and D. Eck. Predicting genre labels for artists using freedb. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[4] Y. Freund and R.E. Shapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[5] B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley, Berkeley, California., 2000.

[6] Marieke Guy and Emma Tonkin. Tidying up tags. *D-Lib Magazine*. online article.

[7] Paul Lamere. Duke Listens! blog at http//blogs.sun.com/plamere/.

[8] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. Position paper, tagging, taxonomy, flickr, article, toread. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, May 2006.

[9] Multiboost, a pure C++ implementation of AdaBoost.MH by N. Casagrande available at http://www.iro.umontreal.ca/~casagran/.

[10] M. J. Salganik, P. S. Dodds, and D. J. Watts. Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, 311(5762):854–856, February 2006.

[11] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.