An empirical investigation of stacking for music tag annotation

Anthony Theocharis
Department of Computer Science
University of Victoria, Canada
Email: anthonyt@uvic.ca

Matt Pierce
Department of Computer Science
University of Victoria, Canada
Email: mpierce@uvic.ca

George Tzanetakis
Department of Computer Science
University of Victoria, Canada
Email: gtzan@cs.uvic.ca

***Abstract*—Automatic tag annotation is one of the most important problems in multimedia information retrieval. It has been motivated by the large amount of unstructured tag annotation data provided by internet users and can be viewed as a variation of multi-label classification with special characteristics and constraints. Stacking is a technique in which the outputs (binary or probabilistic) of a set of binary classifiers (one for each tag) are used as input to a second stage of classification that attempts to exploit latent relationships between tags. This technique (known under a variety of names) has been used in a variety of multimedia tag annotation systems. In this paper we survey these approaches, clarify how stacking system are structured, and empirically investigate stacking using a variety of classifier combinations in the context of tagging pieces of music.**

I. Introduction

Increases in network bandwidth, disk storage and computing speed have made possible the creation of large collections of multimedia objects that can be accessed by anyone with an internet connection. Organizing these large collections for effective retrieval is one of the biggest challenges in multimedia research. The term "tag" refers to any keyword associated to an article, image, video, or piece of music on the web. In the past few years there has been a gradual shift from manual annotation into fixed hierarchical taxonomies to collaborative social tagging where any user can annotate multimedia objects with tags (so called folksonomies) without conforming to a fixed hierarchy and vocabulary. For example, Last.fm is a collaborative social tagging network which collects roughly 2 million tags (such as "saxophone", "mellow", "jazz", "happy") per month [1] and uses that information to recommend music to its users. Another source of tags are "games with a purpose" [2] where people contribute tags as a by-product of doing a task that they are naturally motivated to perform, such as playing casual web games. For example TagATune [3] is a game in which two players are asked to describe a given music clip to each other using tags, and then guess whether the music clips given to them are the same or different.

Tags can help organize, browse, and retrieve items within large multimedia collections. As evidenced by social sharing websites including Flickr, Picasa, Last.fm, and You Tube, tags are an important component of what has been termed as "Web 2.0". The focus of this paper is systems that automatically predict tags (sometimes called autotags) by analyzing multimedia content without requiring any user annotation. Such systems typically utilize signal processing and supervised machine learning techniques to "train" autotaggers based on analyzing a corpus of manually tagged multimedia objects.

There has been considerable interest for automatic tag annotation in multimedia research. Automatic tags can help provide information about items that have not been tagged yet or are poorly tagged. This avoids the so called "cold-start problem" [4] in which an item can not be retrieved until it has been tagged. Addressing this problem is particularly important for the discovery of new items such as recently released music pieces in a social music recommendation system.

From a machine learning perspective automatic tag annotation can be viewed as a variation on multi-label classification. In contrast to traditional classification in which each item is assigned one of k mutually exclusive class labels, in multi-label classification each item can be assigned to multiple labels (tags). Many different approaches to multi-label classification have been proposed in the literature. They leverage feature information computed from a training set of examples annotated with multiple labels to train models that can subsequently be used to predict labels for new examples. There are some unique characteristics and related challenges when the ground truth tag data is obtained from the web. The ground truth training data is *noisy* in the sense that the tags can contain synonyms ("calm" and "mellow"), misspellings ("chello") and hierarchical relations ("symphony" and "classical"). In addition the data is *sparse* meaning that there can be few training examples for a given tag. Finally, the absence of a tag cannot always be taken to mean that the tag is not applicable, as it might be the case that the users have simply not yet considered that tag for the particular multimedia item.

Stacking (or stacked generalization) is a technique in which the output of a first classification stage is used as the input to a second classification stage. In the context of automatic tag annotation stacking consists of training a set of V classifiers (one for each tag) for the first stage. The outputs (binary or probabilistic) of each classifier are then used as a new feature vector to train a second stage of V classifiers. The classifiers in stage 1 are each trained independently and therefore can not take into account relations and dependencies between tags. The second stacking stage attempts to exploit these relations to improve tag prediction.

Although stacking is frequently used in the multimedia literature, the terminology is inconsistent and authors have used a variety of different names for this process. We define stacking more formally, connect it with multi-label classification, and cast previous work using a more consistent terminology. We also empirically investigate different combinations of multi-label classifiers in a stacking architecture using two publicly available datasets for music tag annotation and show that it consistently improves annotation performance.

II. Related Work

Work on associating music with text using audio content analysis and machine learning started as early as 2002 [5], not using tags per se, but using keywords extracted from web-pages that ranked highly in search engine results for particular artists. Around 2007-2008, as social tag annotation became more common, some of the first papers that focused on automatic tag annotation for music started appearing in the literature using different classification approaches and audio feature sets. For example, AdaBoost.MH [6] is used for tag prediction in [7]. A Gaussian Mixture Model over the audio feature space is trained for each word in a vocabulary in the seminal paper by Turnbull et all [8] which also provided the *CAL500* dataset used in this paper and since then has frequently been used to evaluate tag annotation systems. Since then several systems of music tag annotation have been proposed and in some cases evaluated in the Music Information Retrieval Evaluation Exchange (MIREX) [9]. Representative examples include: support vector machines (with stacking) [10], [11], topic models using Latent Direchlet Allocation (LDA) [12], codeword bernoulli average [13], cost-sensitive stacking using Support Vector Machines and AdaBoost [14], and parallel factor analysis [15].

A thorough overview of multi-label classification methods is provided in [16]. They classify approaches to multi-label classification into two major groups. *Problem transformation* methods try to perform multi-label classification by transforming the problem to a series of simpler multi-class or binary classification problems (for which there is a large established literature) and combining their results. *Algorithm adaptation* methods work by adapting/extending existing multi-class and binary classification algorithms to handle the multi-label case. We comment briefly on some of the multi-label algorithms that are particular relevant to this paper and provide more details about their specifics later when describing our experiments. A common problem transformation approach (termed PT4 in [16]) is to train $|L|$ binary classifiers (one for each label l) by using all the instances that contain l as positive examples and all the remaining ones as negative examples. For classification of a new instance x, this method outputs as a set of labels the union of the labels that are output by the $|L|$ classifiers. Another approach mentioned, but not evaluated, in [16] is PT5 in which each example (x, Y) is decomposed into $|Y|$ examples (x, l) with a single label each. The resulting dataset can be used for training any standard multi-class classification algorithm with the added characteristic that the same feature vector will be used in the training sets of several classes that correspond to all the labels for that instance. To classify new instances, a classifier with probabilistic outputs (or more generally a distribution of scores) over all labels in L is then used. In terms of algorithm adaptation methods is straightforward to extend the K-Nearest Neighbor algorithm to handle the multi-label case [17]. The well-known C4.5 algorithm for decision tree learning can also be modified to handle multi-label data by modifying the entropy calculation and allowing multiple labels in the leaves of the decision tree. Adaboost.MH and AdaBoost.MR [6] are two extensions of the AdaBoost algorithm for multi-label classification.

Automatic tag annotation refers to multi-label classification problems in which the labels (tags) are typically obtained socially by users. It can be viewed as a special case of multi-label classification with constraints related to tag sparsity, imbalance between positive and negative examples, and synonymity. Research in automatic tag annotation often originates in the field of multimedia retrieval and, frequently, the connections to multi-label classification are not well understood. In addition to the examples of music tagging mentioned above automatic tagging has also been explored in image [18] and video [19] annotation. An alternative to treating automatic tag annotation as a multi-label classification problem is to treat it as a ranking problem in which, for each new instance, the full tag vocabulary is sorted based on some criterion (examples include AdBoost.MR [6] or PAMIR [20]). In this case a secondary thresholding step can be used to obtain the tags.

Stacking (or stacked generalization) is a term used to describe the process of using the output of a classification stage as an input feature vector to a subsequent classification stage. It has been used in classic multi-class classification [21] but more recently it has also been employed in problems with a more complicated structure, often appearing under different names such as anchor-based classification [22] and semantic space retrieval [23]. To the best of our knowledge, the first use of stacking for multi-label classification was introduced in the context of classifying text documents with multiple labels [24]. In the context of music, tag annotation was originally introduced in [10] and subsequently used in [14], [11]. It also related to the more general techniques of cascaded classification in which the output of several different classification stages are combined in a similar fashion. Cascaded classification models have been used in computer vision for scene understanding [25].

III. Problem formulation

A. Tagging

We consider a tag vocabulary V that consists of $|V|$ unique words (tags). Each tag refers to a semantic concept without any restrictions, for example “rock”, “saxophone”, or “ambient”. The goal of annotation is to find a set $Y = y_1 \dots y_N$ of N words that are semantically meaningful for the particular query audio track. The annotation system is trained using a collection of items (tracks in the case of music) that are each annotated with a set of tags from the vocabulary. Traditional

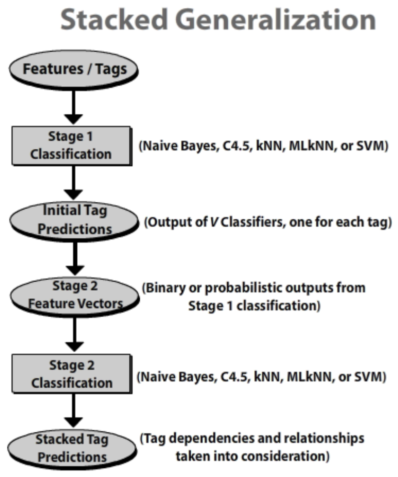
Stacked Generalization
Features / Tags
Stage 1 Classification
(Naive Bayes, C4.5, kNN, MLkNN, or SVM)
Initial Tag Predictions
(Output of V Classifiers, one for each tag)
Stage 2 Feature Vectors
(Binary or probabilistic outputs from Stage 1 classification)
Stage 2 Classification
(Naive Bayes, C4.5, kNN, MLkNN, or SVM)
Stacked Tag Predictions
(Tag dependencies and relationships taken into consideration)

Fig. 1. Block diagram of stacking

single-label classification learns from a set of examples that are each associated with a single label, l, from a set of disjoint labels L, $|L| > 1$. If $|L| = 2$ then the learning problem is called *binary* classification, while $|L| > 2$ then it is called a multi-class classification problem. In *multi-label* classification the examples are associated with a set of labels $Y \subset L$.

B. Stacking

Stacking is a method of combining the outputs of multiple independent classifiers for multi-label classification. The first step of using stacking for multi-label classification is to train $|V|$ individual tag classifiers using a training set $(\mathbf{x}_i, \mathbf{y}_i)$, where x_i denotes the feature vector for instance i and y_i is the associated set of labels. The PT4 or PT5 problem transformation methods, described above, can be used to convert the input training set into training sets appropriate for the individual single-label classifiers. The output of these classifiers (binary or probabilistic) $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{|V|}(\mathbf{x})$ where $\mathbf{x}$ is the input feature vector that is then used as a feature to form a new feature set. Let the new feature set be $z_1, z_2, ...z_{|V|}$. This feature set, together with the original ground truth labels $(\mathbf{z_i}, \mathbf{y_i})$, is then used for training a second stage of stacking classifiers. The goal is to have the stacking classifiers make use of information like the correlation between tags or the accuracy of the first stage classifier to improve the annotation performance. For example suppose that the stage 1 performance for the tag "opera" is not very good but that most of the examples with the tag "opera" receive high probabilities for the tags "classical" and "voice" at stage 1. The stacking stage 2 can take into account this information from other tags and improve annotation performance: something not possible during stage 1, in which each tag is treated independently. Figure 1 shows this process as a block diagram.

IV. Experiments

A. Audio Feature Extraction

Each audio track is represented as a single feature vector. Even though much more elaborate audio track representations have been proposed in the literature we like the simplicity of machine learning and similarity calculation using single feature vectors per audio clip. It has been shown that such song-level features perform quite well [26].

The spectral features used for our experiments are ZeroCrossings, Spectral Centroid, Roll-Off, Flux and Mel-Frequency Cepstral Coefficients (MFCC) (13 coefficients) [27] for a total of 17 features every 20 milliseconds. In addition we compute 14 features related to pitch/chroma every 20 milliseconds. To capture the feature dynamics we compute a running mean and standard deviation over the past M frames:

$$m\Phi(t) = mean[\Phi(t-M+1), .., \Phi(t)] \quad (1)$$
$$s\Phi(t) = std[\Phi(t-M+1), .., \Phi(t)] \quad (2)$$

where $\Phi(t)$ is the original feature vector. Notice that the dynamics features are computed at the same rate as the original feature vector but depend on the past M frames (e.g. M=40, corresponding approximately to a so-called "texture window" of 1 second). This results in a feature vector of 62 dimensions at the same rate as the original 31-dimensional one. The sequence of feature vectors is collapsed into a single feature vector representing the entire audio clip by taking again the mean and standard deviation across the 30 seconds (the sequence of dynamics features) resulting in the final 64-dimensional feature vector per audio clip. Due to space constraints we can not elaborate on the exact details of each feature but they are standard audio features used in music information retrieval computed using the freely available *Marsyas* software framework [1].

B. Datasets

We tested the system on two publicly available audio datasets. The Computer Audition Lab 500 (CAL500) [8] dataset is a selection of 500 Western popular songs recorded by 500 different artists, from between 1958 and 2008. Annotations for the files were collected from different people (a minium of three listeners per track and all using the same predefined vocabulary), and tags are applied in the dataset if more than two people used the same tag for the same song. The dataset includes 174 different tags describing the presence or absence of 33 instruments, 16 vocal characteristics, 47 genres, 18 emotions, 15 preferred listening scenarios and 12 concepts such as tempo or sound quality. For our experiments, we used only 493 of the songs, discarding any that were shorter than 30 seconds. In the resulting dataset, each song is annotated with a minimum of 3 and a maximum of 48 tags. Each tag has a minimum of 5 and a maximum of 437 examples, for a total of 12821 annotations in the dataset. For all tests on the CAL500 dataset, we used 10-fold non-stratified cross-validation.

The Magnatagatune [3] dataset is a collection of 25863 clips of Western music, provided by Magnatune.com and FreeSound.org, that span the genres of classical, new age, electronica, rock, pop, world music, jazz, blues, heavy metal,

[1]http://marsyas.info

and punk. Annotations for the files were collected via the TagATune game-with-a-purpose [], in which two players were asked each to annotate a clip in their own words and then to guess, based on the other player's annotations, whether or not they had been listening to the same song. Tags from this game are applied in the dataset if more than two people independently used the same tag for the same clip. For our experiments, we used 25860 of the songs, discarding any that were shorter than 30 seconds. In the resulting dataset, each song clip is annotated with a minimum of 0 and a maximum of 17 tags. Each tag has a minimum of 23 and a maximum of 4581 examples for a total of 89382 annotations in the collection. For all tests on the Magnatagatune dataset which is much bigger, we used 2-fold non-stratified cross validation.

C. *Classification Algorithms*

In our experiments we have tested various combination of classification algorithms in different stacking configurations. The different configurations are characterized by how the multi-label problem is mapped to single-label problems, the type of classifier used, the output of the classifiers (categorical or probabilistic), and whether stacking was used or not. The following notation is used in the descriptions: X is the set of instances in a dataset; each instance is represented by an m dimensional feature vector; x_i is the feature vector of the ith instance; $x_{i,j}$ is the value of the jth feature of the ith instance.

For all classifiers except the linear support vector machine (SVM), we used the PT4 transformation method [16] in which a classifier is trained for each tag, t, using all songs with that that tag (X_t) as positive examples and all songs without the tag ($X_{\neg t}$) as negative examples. For the linear SVM classifier we utilize the PT5 transformation method. Each multi-label instance is replicated as multiple single-label instances and then a multi-class linear SVM is trained using the one-against-one approach in which binary classifiers for each pair of tags are trained and their predictions are combined to form the multi-class prediction. In order to obtain output class probabilities we utilize the approach suggested by Platt [28].

A standard Naive Bayes (NB) approach is also considered. A separate classifier is trained for each tag. For each dimension, j, of the feature space, two Gaussian distributions are defined: $g_{t,j}$ is defined using the mean and variance of dimension j in X_t, and $g_{\neg t,j}$ is defined in the same way for $X_{\neg t}$. During prediction, the probability that t applies document x_i is predicted as p, where:

$$p = \frac{p_t}{max(p_t, p_{\neg t})} \tag{3}$$

$$p_t = \prod_{j=1}^{m} g_{t,j}(x_{i,j}) \tag{4}$$

$$p_{\neg t} = \prod_{j=1}^{m} g_{\neg t,j}(x_{i,j}) \tag{5}$$

A decision tree (C4.5) is constructed based on the classes t (instances in X_t) and $\neg t$ (instances in $X_{\neg t}$). For each node on the tree, a count is kept of the number of t instances and the number of $\neg t$ instances that are assigned to that node or to its children during training. The probability, p, of tag t applying to instance x_i, is then determined by following the decision tree as far as possible, and taking the ratio of t instances to $\neg t$ instances that reached that node during training. A separate decision tree is trained for every tag t.

A k nearest neighbours (k-NN) classifier ($k = 10$ selected empirically) is trained for every tag t by memorizing each instance in the training set as a point in R^m, described by its feature vector. The probability, p, of tag t applying to instance x_i is then determined by finding k instances in the training set such that the sum of their euclidean distances from x_i is a minimum. p is then equal to the fraction of those k training instances that were from X_t. The ML-kNN classifier [17] differs from the plain kNN classifier only in that, for each tag, the predicted probability of that tag applying is always multiplied by the prior probability of that tag appearing in the training dataset.

For all the classification configurations except SVM we utilized the Mulan java library [2] for multi-label learning which is built on top of the well known Weka machine learning software [3]. The SVM approach is implemented in the Marsyas audio framework and is based on the *libsvm* library [4]. Custom software in Python was written to perform the experiments and collect the evaluation results.

D. *Evaluation*

Evaluation of automatic tagging systems is not trivial. In general the evaluation metrics used are generalizations of commonly used evaluation metrics for single-label classification. A annotated "training" set of instances is used to "train" the classifier and then used to "predict" the tags for a set of instances in a "test" set. We can also distinguish between evaluation metrics that are based on a predicted discrete set of tags (sometimes referred to as the annotation task) and ones that are based on a predicted set of tag affinities/probabilities (sometimes referred to as the ranking task) for each instance in the testing set. A common approach is to treat any classification decision equally and simply count the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) to derive well-known measures such as precision, recall and F-measure. In the case of probabilistic output multiple score thresholds can be considered as possible boundaries for binarization in which case it is common to use Receiver operating characteristics (ROC) curves [29]. An ROC curve is a plot of the true positive rate as a function of the false positive rate. The ROC curve can be summarized by the area under curve (AUC-ROC) which can be found by integrating the ROC curve and is upper bound by 1.0. Random guessing in a retrieval task results in an AUC-ROC of 0.5. Different tag annotation methods can have different operating point characteristics in terms of the tradeoff between true positives and false postives therefore in this paper we mainly use ROC

[2]http://mulan.sourceforge.net

[3]http://www.cs.waikato.ac.nz/ml/weka

[4]http://www.csie.ntu.edu.tw/~cjlin/libsvm

TABLE I
CAL500: AVERAGE TAG AUC-ROC FOR DIFFERENT STACKING CONFIGURATIONS

	Stage2					
Stage 1	None	C4.5	kNN	MLkNN	NB	SVM
RND	0.487	0.359	0.461	0.382	0.441	0.407
RND(Priors)	0.424	0.366	**0.457**	0.382	**0.438**	0.414
C4.5	0.404	**0.444**	**0.558**	**0.523**	**0.570**	-
kNN	0.568	0.392	0.558	0.480	**0.626**	**0.579**
MLkNN	0.475	0.384	**0.553**	**0.482**	**0.597**	**0.554**
NB	0.619	0.443	0.565	0.486	0.545	0.567
SVM	0.592	0.433	0.584	0.504	**0.619**	**0.638**
ORACLE	-	0.902	0.850	0.780	0.931	0.981

TABLE II
MAGNATAGATUNE: AVERAGE TAG AUC-ROC FOR DIFFERENT STACKING CONFIGURATIONS

	Stage2					
Stage 1	None	C4.5	kNN	MLkNN	NB	SVM
C4.5	0.347	0.325	0.297	**0.412**	**0.718**	-
kNN	0.535	0.397	0.504	**0.478**	**0.790**	-
MLkNN	0.520	0.407	0.499	0.478	**0.789**	-
NB	0.728	0.344	0.480	0.459	0.613	**0.793**
SVM	0.819	0.399	0.541	0.520	0.701	**0.839**

and AUC-ROC to investigate the performance of stacking configurations as they are more general. A final complication is that calculating metrics over the entire set of tags can be misleading as good performance on "popular" tags that appear in many instances will dominate but typically a more balanced response where all tags are considered is desired. In order to address this concern we also consider evaluation metrics averaged across tags. Finally it is important to note that in most cases evaluation metrics based on annotated ground truth underestimate what the true performance of the system would be if evaluated by humans [12]. The reason is that frequently, predicted tags that humans would consider applicable are not present in the ground truth and therefore evaluated as mistakes.

E. Results

Tables I,II,III,IV show the AUC-ROC values for different stacking configurations. Both the overall as well as the values avearaged across tags are considered. As a baseline we consider the performance of a model that randomly assigns tags (RND) as well as a model that randomly assigns tags using the prior probabilities of each tag in the training set (RND(priors)). The first column in each table shows the type of classifier used

TABLE III
CAL500: OVERALL AUC-ROC FOR DIFFERENT STACKING CONFIGURATIONS

	Stage2					
Stage 1	None	C4.5	kNN	MLkNN	NB	SVM
RND(Priors)	0.600	**0.525**	**0.749**	**0.808**	**0.747**	**0.810**
RND	0.496	**0.516**	**0.755**	**0.808**	**0.764**	**0.808**
C4.5	0.662	0.627	**0.777**	**0.808**	**0.756**	-
kNN	0.802	0.594	0.795	**0.825**	0.743	**0.833**
MLkNN	**0.828**	0.573	0.793	0.825	0.738	0.828
NB	0.667	0.657	**0.795**	**0.825**	0.661	**0.831**
SVM	0.844	0.593	0.810	**0.835**	0.697	**0.852**
ORACLE	-	0.995	0.942	0.967	0.942	0.972

TABLE IV
MAGNATAGATUNE: OVERALL AUC-ROC FOR DIFFERENT STACKING CONFIGURATIONS

	Stage2					
Stage 1	None	C4.5	kNN	MLkNN	NB	SVM
C4.5	0.741	**0.788**	0.419	**0.850**	**0.742**	-
kNN	0.757	**0.695**	0.718	**0.883**	**0.849**	-
MLkNN	0.892	0.695	0.715	0.881	0.847	-
NB	0.762	0.760	0.712	**0.874**	0.666	**0.913**
SVM	0.923	0.666	0.761	0.894	0.743	**0.933**

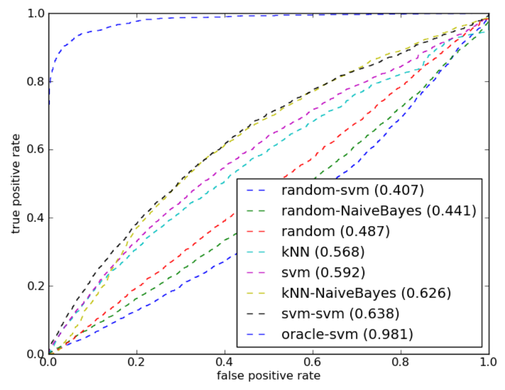

Fig. 2. Receiving Operating Characteristic Curves for different stacking confgigurations

for stage 1 and the column headers correspond to the type of stacking classifiers used for stage 2. The first column of AUC-ROC results corresponds to no stacking. Entries in bold indicate stacking configurations in which stacking improves the results compared to just using stage 1. For some entries marked with a - we encountered numerical problems in our simulations and we were not able to obtain results. We have been investigating the cause but have not been able to find it although we suspect it has to do with insufficient variance in the stage 1 affinities interfering with the training of stage 2 SVM classifiers. In order to establish an upper bound to stacking performance we also consider an "ORACLE" configuration in which the actual ground truth labels in the test set are used as input to a stacking classifier trained on the ground truth labels of the training set. As can be seen the random configurations perform close to the theoretical 0.5 AUC-ROC for random single tag classification except in the overall AUC-ROC where the use of random priors improves the result slightly (0.6). Also, one can see that stacking using the theoretical "ORACLE" upper bound performs very well, close to the ideal AUC-ROC of 1 of perfect classification; difference between the training and testing set used in cross-validation account for the deviation. The combination of the particular features and classifiers are closer to the lower bound than the upper bound which indicates there is a lot of progress that can be made with discovering new features for music tagging. However, several of the stage 1 classifiers peform better than random and, in general, stacking has a positive effect on performance, as can be seen by all the bold entries.

Notice that stacking also benefits if the tags have been assigned randomly in stage 1 as the relations between tags are still modeled based on the ground truth in training set (i.e the prior probabilities and joint probabilities among the tags). This influence of stacking in the RND configurations is only in the overall AUC-ROC.

One somewhat unexpected result was the excellent performance of the Naive Bayes classifiers as a stacking stage 2 classifier. We believe this might be due to the better fit of the NB classifier to a probabilistic formulation. Given the simplicity and speed of training it can be an easy addition to existing music and more generally multimedia tagging systems. In both the average and overall AUC-ROC the combination SVM for both stage 1 and stage 2 resulted in the best results for both datasets. The use of prior probabilities in ML-KNN compared to KNN shows an effect in the overall configuration but does not make any difference in the average over tags. Figure 2 shows some representative ROC curves for the best two stacking configurations in CAL500 as well as the upper and lower bounds of random and oracle tag ranking.

V. Conclusion

In this paper we established conections and correspondances between the literature in multimedia information retrieval, multi-label classification and stacking which is a technique in which the output of a classification stage is used as input to a subsequent classification stage. We investigated the performance of stacking using a variety of classifier configurations using two dataset for music tag annotation. Based on experimental results stacking seems to be a valid strategy for improving the performance of music tag annotation systems. The simple Naive Bayes classifier and the Support Vector Machine (SVM) seem to be the best performing choices both as stage 1 classifiers and as stacking classifiers in stage 2. The effect of stacking is more noticeable when the performance in stage 1 is poor. There are many directions for future work that we plan to investigate. We would like to contrast stacking both in terms of annotation and retrieval performance and computational requirements with ranking approaches as well as other alternatives to multi-label classification such as AdaBoost.MH and AdaBoost.MR as well as topic models. The performance of stacking in the presence of synonyms and hierarchical relations can also be investigated by creating synthetic examples exhibiting such effects. We also believe stacking should also be applicable to other areas of multimedia information retrieval such as image and video annotation and plan to investigate this in the future.

References

[1] P. Lamere, "Social tagging and music information retrieval," *Journal of New Music Research*, vol. 37, no. 2, pp. 101–114, 2008.

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

[3] E. L. M. Law, L. V. Ahn, R. B. Dannenberg, and M. Crawford, "Tagatune: A game for music and sound annotation," in *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, 2007.

[4] A. Schein, A. Popescul, L. Ungar, and D. Pennock, "Methods and metrics for cold-start recommendations," in *Proc. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2002.

[5] B. Whitman and R. Rifkin, "Musical query-by-description as a multiclass learning problem," in *In Proc. IEEE Multimedia Signal Processing Conf. (MMSP)*, 2002, pp. 153–156.

[6] R. Shapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–68, 2000.

[7] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green, in *Adv. in Neural Information Processing Systems*.

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

[9] S. J. Downie, "The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research," *Acoustical Science and Technology*, vol. 29, no. 4, pp. 247–255, 2008.

[10] S. R. Ness, A. Theocharis, G. Tzanetakis, and L. G. Martins, "Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs," in *Proc. ACM Multimedia*, 2009.

[11] K. Seyerlehner, Widmer, G. Schedl, and P. M. Knees, "Automatic music tag classification basd on block level features," in *Sound and Music Computing*, 2010.

[12] E. Law, B. Settles, and T. Mitchell, "Learning to tag from open vocabulary labels," in *Principles of Data Mining and Knowledge Discovery*, 2010, pp. 211–226.

[13] P. C. M. Hoffman, D. Blei, "Easy as cba: A simple probabilistic model for tagging music," in *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, 2009.

[14] H.-Y. Lo, J.-C. Wang, H.-M. Wang, and S.-D. Lin, "Cost-sensitive stacking for audio tag annotation and retrieval," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[15] Y.Panagakis and C.Kotropoulos, "Automatic music tagging via parafac2," in *Proc. IEEE Int. Conf. Audio, Speech and Signal Processing*, 2011, pp. 481–484.

[16] G. Tsoumakas and I. Katakis, "Multi label classification: An overview," *Int. Journal of Data Warehouse and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[17] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification." in *Granular Computing*, X. Hu, Q. Liu, A. Skowron, T. Y. Lin, R. R. Yager, and B. Zhang, Eds. IEEE, 2005, pp. 718–721.

[18] C.-F. Tsai and C. Hung, "Automatically annotating images with keywords: A review of image annotation systems," *Recent Patents on Computer Science*, vol. 1, pp. 55–68, 2008.

[19] G. Toderici, H. Aradhye, M. Pasca, L. Sbaiz, and J. Yagnik, "Finding meaning on youtube: Tag recommendation and category discovery," in *CVPR'10*, 2010.

[20] D. Grangier and S. Bengio, "A discriminative kernel-based approach to rank images from text queries," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1371–1384, 2008.

[21] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[22] A. Berenzweig, D. P. W. Ellis, and S. Lawrence, "Anchor space for classification and similarity measurement of music," in *Proc. of Int. Conf. on Multimedia and Expo (ICME)*, 2003, pp. 29–32.

[23] M. Slaney, "Semantic-audio retrieval," in *Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.

[24] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2004.

[25] G. Heitz, S. Gould, A. Saxena, and D. Koller, "Cascaded classification models: Combining models for holistic scene understanding," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2009.

[26] M. Mandel and D. Ellis, "Song-level features and support vector machines for music classification," in *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, 2005.

[27] G. Tzanetakis and P. Cook, "Musical Genre Classification of Audio Signals," *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 5, Jul. 2002.

[28] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*. MIT Press, 1999, pp. 61–74.

[29] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.