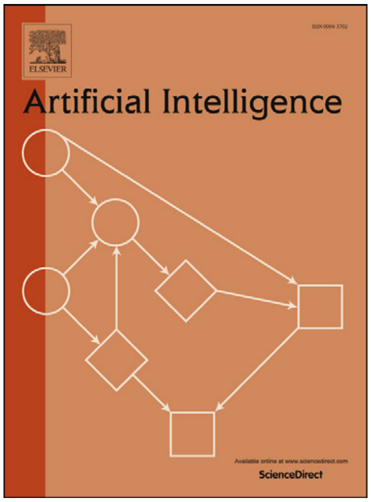
# Learning Pseudo-Tags To Augment Sparse Tagging In Hybrid Music Recommender Systems

Ben Horsburgh, Susan Craw, Stewart Massie



PII: S0004-3702(14)00136-2
DOI: 10.1016/j.artint.2014.11.004
Reference: ARTINT 2799

To appear in: *Artificial Intelligence*

Received date: 25 March 2013
Revised date: 12 November 2014
Accepted date: 18 November 2014

# Learning Pseudo-Tags To Augment Sparse Tagging In Hybrid Music Recommender Systems

Ben Horsburgh, Susan Craw[*], Stewart Massie[*]

*School of Computing Science & Digital Media and IDEAS Research Institute*
*Robert Gordon University, Aberdeen, UK*

**Abstract**

Online recommender systems are an important tool that people use to find new music. To generate recommendations, many systems rely on tag representations of music. Such systems however suffer from tag sparsity, whereby tracks lack a strong tag representation. Current state-of-the-art techniques that reduce this sparsity problem create hybrid systems using multiple representations, for example both content and tags. In this paper we present a novel hybrid representation that augments sparse tag representations without introducing content directly. Our hybrid representation integrates pseudo-tags learned from content into the tag representation of a track, and a dynamic weighting scheme limits the number of pseudo-tags that are allowed to contribute. Experiments demonstrate that this method allows tags to remain dominant when they provide a strong representation, and pseudo-tags to take over when tags are sparse. We show that our approach significantly improves recommendation quality not only for queries with a sparse tag representation, but also those that are well-tagged. Our hybrid approach has potential to be extended to other music representations that are used for recommendation, but suffer from data sparsity, such as user profiles.

*Keywords:* Music Recommendation, Hybrid Representations

[*]IDEAS Research Institute, Robert Gordon University, Garthdee Road, Aberdeen AB10 7GJ, UK, Email: s.craw@rgu.ac.uk, Tel: +44 1224 262711

*Email addresses:* s.craw@rgu.ac.uk (Susan Craw), s.massie@rgu.ac.uk (Stewart Massie)

## 1. Introduction

During the past decade a massive shift has occurred in the way that people store, listen to and discover music. Traditionally, either a single or album was stored on external media such as cassette or CD. With advances in hardware and the growth of the world wide web, this storage has moved to large music collections on personal hard-drives and websites offering massive catalogues of music.

An interesting new problem that has arisen as a consequence of this shift, is how to find and discover new music. Radio stations and music magazines continue to offer advice but, with such a large amount of music readily available, many people now look to recommender systems provided by online services. Users provide a query, such as an example track, and a recommender system will then return a set of recommendations to the user.

Core to this task of providing recommendations is the representation of tracks. Each track in the collection must have a corresponding representation, which describes the key facets of the track. Many state-of-the-art music recommender systems make use of tag-based representations, for example the hugely popular Last.fm service [1]. The tags typically consist of two components: standardised meta-data, such as artist, track title and genre; and free-text that users provide. Free-text tags are extremely useful to a recommender system, providing a wealth of different information, such as genres, artists, topics, opinions and contexts [2, 3]. Many of these tags provide useful information for recommendation, for example, tags relating to genres, artists and topics. Context such as location can be used to improve personalised recommendations, such as when it is known a user is working in a given location [4, 5].

Uninformative tags can be problematic for tag-based recommender systems, but a far larger problem is when very few tags exist at all. This is a problem commonly known as cold-start and arises from a large popularity bias in music listening habits [6]. As a result, many tracks are under tagged; a track is tagged if it is listened to, but a track is only recommended, and therefore listened to, if it already has tags.

Figure 1 displays the general pattern of results found in the experiments, and here is used to illustrate the cold-start problem in query-by-example tag-based recommender systems. The horizontal axis shows the number of tags a query has, and the vertical axis shows recommendation quality. The dotted curve in Figure 1 represents average recommendation quality in a tag-

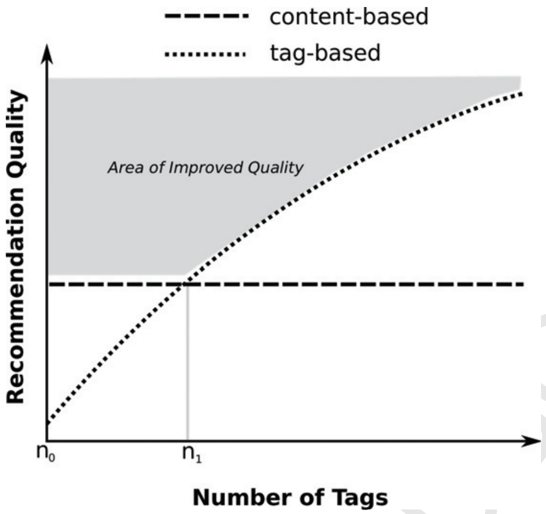


Figure 1: The Cold-Start Problem

based recommender system. As the number of tags increases, so does the quality of tag-based recommendation, since a larger amount of meaningful knowledge is available. The horizontal dashed line in Figure 1 represents average recommendation quality in a content-based recommender system. Since this system does not rely on tags, the quality of recommendations does not vary as the number of tags changes. For a low number of tags, content-based recommendations are better than tag-based, but as tags increase the tag-based system becomes better than content.

A simple hybrid approach, which can reduce the effects of the cold-start problem, is to use content-based methods for low tagged queries, and then switch to tag-based methods. This hybrid does not offer an improvement over any individual method, but will produce better recommendations as an overall system. In this simple hybrid approach however, deciding when to change from content- to tag-based recommendation can cause problems.

Ideally, the switch from content- to tag-based recommendation is made at point $n_1$ in Figure 1. In this case, the maximum performance is achieved for all numbers of tags. However, in practice it is very difficult to get this switch over correct. If the switch is made too early then performance of the system is decreased by not using the strongest representation. Similarly, if the switch is made too late, then the full potential of the tag-based recommender is not taken advantage of. Further, the extreme case of cold-start is at point $n_0$, where a query track has no tag representation. This does not affect content-based recommendation, but will reduce a tag-based system to making random recommendations.

The aim of a hybrid system designed to reduce cold-start is to consistently provide recommendations in the shaded area. In this case, the hybrid system is able to take advantage of both content and tags, and through their combination improve recommendation quality in all cases. Making recommendations in this area is the motivation for developing our approach to cold-start recommendation using a hybrid recommender system.

In this paper related work on the cold-start problem, hybrid recommender systems, and their constituent parts is discussed first. We then define and present our approach to learning pseudo-tags from content, which guarantees a tag-based representation for every track in a collection. Next, we present our novel hybrid representation, which is a dynamic combination of both pseudo-tags and tags. To evaluate our approach, we then examine both the recommendation quality, and the quality of pseudo-tagging. Finally, we present our results and conclusions.

## 2. Related Work

Tag-based representations, and the knowledge they describe, are very important to music recommender systems. For certain recommendation problems, tags are able to offer greater performance than either content-based or collaborative filtering methods [7]. Whilst a lot of progress has been made in how this knowledge can best be used, cold-start continues to cause problems in many tag-based systems [2].

To try and understand why cold-start remains a problem, Turnbull et al. [8] compare five approaches to collecting tags, in terms of both scalability and quality. The two approaches they discuss that are relevant to our work are social tagging and auto tagging. Social tagging systems allow users to label tracks with any text, and allows the representation to benefit from social wisdom and context. The main weakness the authors highlight however is that social tags tend to be sparse for unpopular tracks, a phenomenon often referred to as the long tail.

Investigating the level of tagging among various websites, the authors of [9] confirm that this long tail is present in each. This is further illustrated for music tags by the Last.fm contribution to the million song dataset [10, 11]. To compound the problem further, the authors of [6, 12] also find evidence of popularity bias in music listening habits. The combination of the long-tail in tagging, and popularity bias in listening habits, results in many tracks being very under tagged; the cold-start problem.

Two simple approaches to reducing cold-start in tag-based representations are to collect them by survey [13] or through an online game [14, 15]. Two of the most successful approaches are the Tag-a-Tune game [15] and MajorMinor game [16], collecting tags for $30,237$ and $2,300$ tracks respectively. Both of these approaches however suffer from scalability problems, and as such risk limiting tagging to popular tracks, rather than tagging the popularity long-tail where it is needed.

A further approach to reducing cold-start in tag-based recommender systems is auto-tagging [17]. The goal of auto-tagging is to learn a set of tags which are relevant to a track; these tags may then be used as a representation of the track. In general, most auto-tagging systems follow a similar approach; the content of tracks is modelled, and a training set of tags created. A multi-class classification algorithm is then used to predict the top $n$ most likely classes, or tags, for any given track.

To generate auto-tags, the content of each track is modelled from content-based representations. The most common content-based representations used include chroma, rhythm and texture [18], which are typically described as a bag-of-features. For auto-tagging, a growing trend has been to model these bags-of-words using Gaussian Mixture Models (GMM) [13], which describe the bag-of-words as a mixture of Gaussian distributions. Tags may be modelled in a similar manner, in which case auto-tagging can be achieved using the weights of correlated distributions. This method achieved the top f-measure and AUC ROC-tag score in the MIREX (Music Information Retrieval Evaluation eXchange) 2008 auto-tagging competition [19].

A further GMM based approach was adopted by Tardieu et al. in their MIREX 2011 auto-tagging submission [20]. A Universal Background Model (UBM) GMM is trained using the expectation maximisation algorithm. The model for each track is then learned by adapting the UBM to fit the track content data. This is achieved by modifying only the mean vectors of the UBM, and thus each track can be represented by the mean vectors of its Gaussian components. In MIREX 2011, this method achieved the top tag classification accuracy, negative example accuracy, precision, and f-measure.

The authors in [21] argue that such classification approaches to auto-tagging are too simple, and that further statistical modelling is required. They refine the approach by modelling concepts, not tracks, using GMMs.

These models are then refined using a set of SVM classifiers. The approach is competitive with other methods, but does not justify the extra computational complexity that is introduced.

We take inspiration for our work from a more simple approach, which has been used for style and mood annotation [22, 23], and for propagating semantic information [24]. The authors in each case use content-based retrieval to provide a set of ordered retrievals. It is the order and number of these retrievals which should be used for annotation, not tag correlations with content. This general idea has been used successfully in other related domains, including playlist generation [25], and image tagging systems [26].

Further approaches to reducing cold-start make use of a hybrid system, many of which rely on content, not auto-tags. In our previous work we investigated generalising concatenated tag-content representations [27]. Using this generalisation we were able to increase the cold-start discovery of tracks, while maintaining a recommendation quality comparable to a tag-based recommender system. A similar hybrid approach was investigated by Levy and Sandler [28], but using a representation they call muswords. In this hybrid system the authors first cluster the content representation so that it is comparable with tags, and then concatenate the representations.

This notion of combining musical representations, and then generalising them is a growing trend in hybrid recommendation systems. Combinations of representations include: audio features and user-ratings [29]; audio symbols and user-ratings [30]; user-tags and user-ratings [31]; and tags and audio [27, 28]. The intuition behind such approaches is that sparsities in some representations (tags, user-ratings) can be overcome using content. In each case, the way that content is used is based on its correlation with another representation. One issue with these approaches however is that content does not always correlate with other representations. For example, tags can describe non musical properties, and so looking for non-existent correlations may introduce noise into the hybrid system.

This problem of correlating content with other representations is avoided in [31]. The authors look for correlations in user-tags and user-ratings; the user provides a direct link and so correlations are meaningful. To reduce the sparsity, and therefore cold-start problem, the authors investigate propagating user-tags based on content-based similarity. The authors describe this as a way of combining content with other features, which slightly obscures the insight of their method.

It is known that content on its own does not perform well compared to other methods [27]. Based on the previous work in hybrid recommender systems, we propose that content should not be used for recommendation directly at all. Content should be used to strengthen other representations, but should not be integrated into these representations. Our approach attempts to follow these lessons from auto-tagging and hybrid recommendation, and we propose a method where content is used to define a pseudo-tag representation, which is then used in our hybrid recommender system.

## 3. Constructing a Pseudo-Tag Representation

Methods designed to address the cold-start problem in music representations may make use of audio content. The primary advantage of such content-based approaches is that for many recommender systems the audio for every track is available. However, recommendations based solely on content are not as strong as when tags are used. One reason for this is that content is unable to capture the rich social and contextual aspects that may be contained in tags.

To reduce the problems encountered when using a content-only method for cold-start representations, we have developed a novel approach to learning a pseudo-tag representation. Pseudo-tags are tags that are assigned to a track algorithmically, based on the content representation of the track. Further, pseudo-tags should be able to describe non-musical properties of a track, based on the tags that similar tracks are assigned. Thus, pseudo-tags offer a representation that is available to every track in a collection, and that is more knowledge rich than content alone. In this section we first describe the content representation we use, and then our method for learning pseudo-tags.

### *3.1. Representing Audio Content*

There are several different ways that musical content can be described. For example, musical aspects such as chroma and melody may be captured, or structural elements like beats and sections may be described. One of the most widely-used descriptors for music recommendation however is texture, also known as timbre [6].

To describe texture, we use the Mel-Frequency Spectrum (MFS) representation developed in previous work [32]. The MFS representation is a musical adaptation of the well-known Mel-Frequency-Cepstral-Coefficients (MFCC) texture representation, which was initially developed for speech recognition [33].

The first part of extracting the MFS representation from a track's audio is similar to many audio representations. The track is split into windows and for each window the time-domain audio signal is converted into a frequency-domain spectrum using a fast Fourier transform (FFT). This spectrum describes frequency in terms of equally spaced bins, where each bin $k$ represents the contribution of that frequency range to the original time-domain signal. The specific frequency $f$ that bin $k$ describes is

$$f = \frac{k}{NT} \tag{1}$$

where $N$ is the number of samples that describe each frame and $T$ is the length of each frame in seconds. For our experiments, the audio sampling rate is 44.1kHz and the track is split into windows of length $2^{13}$ (186ms). The audio signal is converted into a frequency-domain spectrum of size $2^{12}$ with maximum frequency 22.05kHz and bin resolution 5.4Hz.

To create the MFS texture representation from the frequency spectrum obtained, the spectrum bins $k$ must be grouped into buckets according to the mel scale [34]. This scale describes the way in which humans perceive frequency, and is roughly linear up until 1000Hz, and then becomes logarithmic. The following equation describes the conversion from frequency $f$ to mel $\phi(f)$.

$$\phi(f) = 2595 \log_{10}\left(\frac{f}{700} + 1\right) \tag{2}$$

Triangular filters are used to discretise the frequency-spectrum into mel-spaced buckets, as illustrated in Figure 2. Each filter is of equal width in terms of the mel scale, defined as

$$\Delta(\phi) = \frac{\phi(f_{\max})}{M} \tag{3}$$

where $M$ is the number of filters used. The lower bound $l$ and upper bound $u$ frequency spectrum bins of each filter $m$ are therefore calculated as

$$l_m = \phi^{-1}(m\Delta\phi)NT \tag{4}$$

$$u_m = \phi^{-1}((m+1)\Delta\phi)NT \tag{5}$$

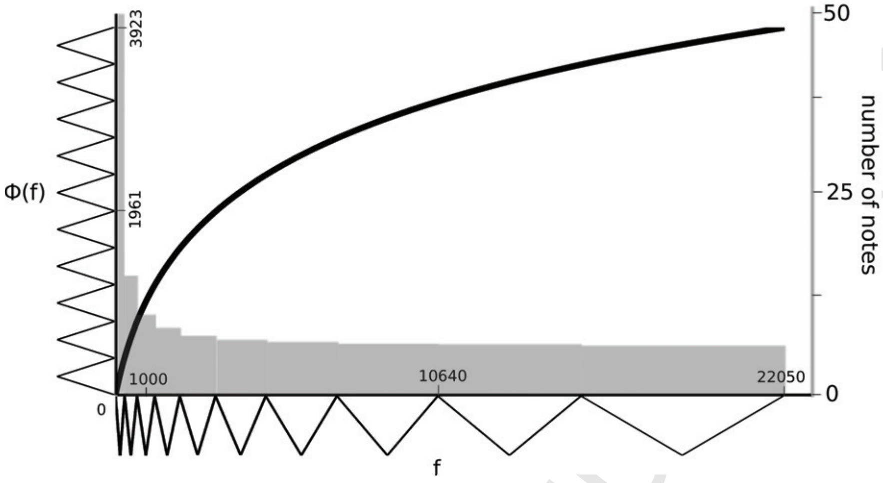


Figure 2: Frequency-mel relationship and filter placement

and thus the width of filter $m$ in terms of frequency spectrum bins $k$ is

$$\Delta k_m = u_m - l_m \tag{6}$$

From these lower and upper bound frequency dimensions, a triangular filter bank $F$, where each column contains a single triangular filter, may be constructed to achieve the discretisation as

$$F_{k_m}(x) = \begin{cases} \frac{2}{\Delta k_m + 1} \cdot \left( \frac{\Delta k_m + 1}{2} - \left| (x - l_m) - \frac{\Delta k_m - 1}{2} \right| \right) & \text{if } l_m \leq x \leq u_m \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and finally applied to the frequency spectrum $X$ as $Y = XF$.

Each filter is spaced equally along the mel scale, shown on the vertical axis of Figure 2, therefore distributing the buckets along the frequency axis as shown by the horizontal axis. The curve shown in Figure 2 is the mel scale defined in Eq. (2). One further consideration in describing musical texture is how notes are distributed amongst this discretisation. The number of musical notes in each bucket can be calculated as

$$\text{notes}(f, f_{\text{ref}}) = 12 \log_2\left(\frac{f}{f_{\text{ref}}}\right) \tag{8}$$

where $f_{\text{ref}}$ is a reference frequency, which we set as the note $C_0$, 16.35Hz. The results of this equation are illustrated by the bars in Figure 2, showing most musical notes occur at low frequencies, and very few occur at higher frequencies. This observation affects the way that the frequency range may be discretised when constructing the MFS representation, where changing the number of triangular filters used can have a significant effect. Tuning this parameter also requires an understanding of how the representation will be used in a retrieval model. For the experiments 40 mel-spaced filters are used, based on empirical evidence found to be best for MFS with aggregation-based recommender models [32].

The vector created as a result of discretising the frequency range using mel-spaced filters is the MFS representation; one vector for each time sample. The collection of vectors for each track therefore is the input to any model which must use them.

### *3.2. Content-Based Recommendation*

After obtaining an MFS vector for each window in a track, the next step is to model these vectors. Based on previous work in [32] we construct a single MFS vector for the track, where each dimension in this vector is the mean value of that dimension across all the track's MFS vectors.

We compute the mean MFS vector for each track in our collection, and construct a mean-vector by track matrix. Latent semantic analysis [35] is then used to discover musical texture concepts from the mean-vector by track matrix M such that

$$M = U\Sigma V^T \tag{9}$$

Using this model we now have a baseline method for cold-start recommendation. Given a query track $q$, the track's MFS vector can be projected into our LSA model as

$$q' = \Sigma^{-1}U^T q \tag{10}$$

and recommendations made using Cosine similarity with $k$ nearest neighbour retrieval. This baseline approach will be referred to as **Content**.

### *3.3. Pseudo-Tag Recommendation*

The `Content` model described can be used not only to provide cold-start recommendations directly, but also to generate pseudo-tags for a track. These pseudo-tags are tags which are learned offline, and may be used to represent a track. Pseudo-tags are not intended to be revealed to users as tag suggestions, but are instead used as a replacement representation for content. It is expected that pseudo-tag representations will benefit over content because factors such as context and opinions will also be present in the representation as the result of human tagging activity. This generation of pseudo-tags allows for recommendations to be made using tags directly, rather than using content for recommendation.

The first step to generating pseudo-tags is to find tracks that are similar to a track $\mathcal{T}$. First, a nearest-neighbour retrieval is made using content-based similarity, where the $K$ most similar tracks to $\mathcal{T}$ are retrieved. Each of these $K$ retrieved tracks will have an associated tag representation, which can then be used to learn pseudo-tags for track $\mathcal{T}$.

A simple approach to creating a pseudo-tag vector is to treat all tag vectors as being binary, and count tag occurrences. One drawback of using a binary representation is that the original contribution of each tag is not fully taken into account. A more refined approach is to weight the contribution of each retrieved track's tag vector, based on how similar it is to the query. However, content-based similarity can vary depending on the query, and so each neighbour's rank should be used, as shown by [39, 40]. We therefore propose a solution where a neighbour's rank is used as a weighting function to the original tag contribution. The weight $w$ of a given track's tag vector is determined as

$$w_k = 1 - \frac{k-1}{K} \tag{11}$$

where $k$ is the position of the track in the nearest neighbour retrieval in the range 1 to $K$, and $K$ is the number of nearest-neighbours retrieved.

The pseudo-tag vector $p$ for a given query track is initialised to 0, and the dimensions corresponding to each nearest neighbour's tags calculated as

$$p_i = \sum_{k=1}^{K} w_k t_i(k) \tag{12}$$

where $t_i(k)$ is the term-frequency of the $i$th tag in the tag vector of the track at position $k$ in the nearest neighbours. Tag vectors are sparse vectors of the tag vocabulary, and pseudo-tag vectors are aggregations of tag vectors.

Given this pseudo-tag representation, we are now able to recommend tracks using only tag-based methods, where the pseudo-tag vector is used as a substitute for the query's mean MFS vector. This approach will be referred to as **PseudoTag**.

While investigating the weightings for Eq. (12), we looked at introducing thresholds for the nearest neighbour search. One intuition was that a minimum threshold value should be placed on the number of tags that each nearest neighbour has. For example, only including tracks with more than 20 tags in our nearest neighbour search. It was found however that such a threshold is harmful, and that all tracks should be allowed to contribute.

## 4. Tag and Pseudo-Tag Hybrid

Pseudo-tags can be used to improve the quality of cold-start recommendations, but they do not solve one other important problem; when should pseudo-tags be used? A threshold may be set based on the number of tags a query has, and a decision made as to whether or not the track is a cold-start item. If this threshold is too high however then recommendation quality is hampered by the pseudo-tags; if the threshold is too low then recommendation may still be hampered by a lack of useful tags.

To overcome this problem we develop a novel approach to creating a dynamic hybrid representation, using both tags and pseudo-tags, which does not rely on this kind of threshold. In this section we first discuss the selection of how many pseudo-tags to use, and then our weighting and retrieval method.

### *4.1. Selecting Pseudo-Tags*

Pseudo-tags can be learned using Eq. (11) and Eq. (12). In this pseudo-tag vector many tags will be present, since it is the summation of $K$ tracks' tag vectors. This increased number of pseudo-tags over the typical number of tags can provide problems when computing similarity between a pseudo-tag vector and tag vector, and so only the most relevant pseudo-tags should be selected.

Pseudo-tag and tag vectors may be compared using cosine similarity

$$\text{similarity}(p, t) = \frac{p \cdot t}{||p||\,||t||} \tag{13}$$

which compares the intra-vector strength of each dimension in vector $p$ and $t$ respectively. The denominator normalises each vector, so that it is effectively a unit vector, and therefore also normalises the weights of each dimension.

In most cases where cosine similarity is used this normalisation is desirable. However, in a recommender system containing both tag and pseudo-tag representation this introduces a bias; tag-based representations will tend to be similar to other tag-based representation, and pseudo-tag representations with tend to be similar to other pseudo-tag representations. The reason for this is that a tag-based representation will tend to have a few strong tags, and a moderate number of weak tags. In comparison, a pseudo-tag representation is likely to have many strong tags, and many weak tags. In a hybrid representation which contains both pseudo-tags and tags, this means that well-tagged queries are unlikely to generate cold-start recommendations, and conversely cold-start queries are unlikely to generate well-tagged recommendations.

To overcome this bias we merge the pseudo-tag and tag vectors for each track into a new vector, and an important step in this process is first restricting the number of pseudo-tags which a track may have. As the number of tags in a track increases, so does the recommendation quality, and so, intuitively, as the number of tags $T$ increases, the number of pseudo-tags $P$ should decrease. If too many pseudo-tags are introduced to a well-tagged track's representation, then the existing tags will not influence recommendation as much as they should. In this combination, both the pseudo-tag and tag vectors must both be normalised to a unit-vector before combination, ensuring that one does not overpower the other.

To test this intuition we evaluate recommendations using differing numbers of pseudo-tags and tags in a combined representation. Using a leave-$n$-out strategy on our training set, we evaluate recommendation quality using different numbers of pseudo-tags, and aggregate these results based on how many tags a track has. To provide a meaningful aggregate, the number of tags each track has are discretised into buckets of 10 tags, for example, all tracks with between 1 and 10 tags are aggregated together.

The solid line in Figure 3 illustrates the number of pseudo-tags, which when combined with a tag vector produce the best recommendation quality. The dashed line is a line-of-best-fit through this data. Based on this observation, we therefore limit the number of pseudo-tags representing a track using the line-of-best-fit. Pseudo-tags are ranked by the weighting they are given using Eq. (12), and the top $P$ pseudo-tags are selected, based on the line in

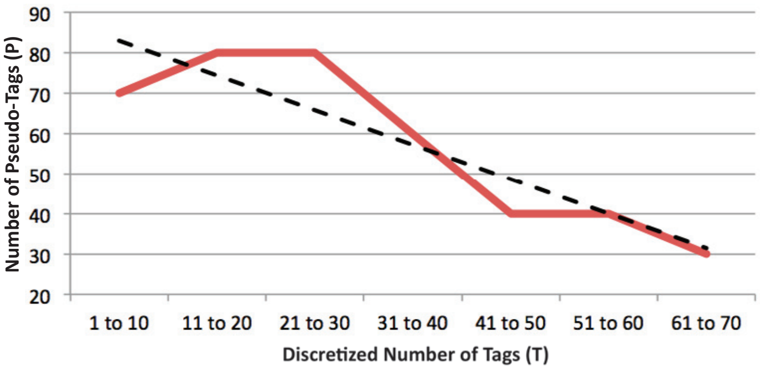


Figure 3: Balancing number of pseudo-tags against tags

Figure 3, as the `PseudoTag` representation to be used in our hybrid model.

### *4.2. Weighting Tags and Pseudo-Tags*

Having obtained a suitable pseudo-tag vector, the next step is to combine the pseudo-tag and tag vectors into a single representation. This is achieved by assigning both the pseudo-tag and tag vectors an importance weighting, and then summing the vectors. Thus, the hybrid vector $h$ is computed as

$$h_i = \alpha p_i + (1 - \alpha) t_i \tag{14}$$

where $p$ is the pseudo-tag vector, $t$ is the tag vector. Both $p$ and $t$ are normalised per-track prior to computing the hybrid vector. The $\alpha$ parameter weights the influence of pseudo-tags and tags.

From our previous evaluation we found that the weight of pseudo-tags $\alpha$ must also decrease linearly, illustrated in Figure 4. The solid line illustrates the weight of pseudo-tags which produce the best recommendation quality. The dashed line is a line-of-best-fit through the data.

We therefore weight the pseudo-tag vector as

$$\alpha = ln + c \tag{15}$$

where $l$ is the gradient reducing the pseudo-tag weight, set empirically as $l = -0.005$, and $c$ is the maximum weight that pseudo-tags may have, set as $c = 0.5$. This value for $c$ is chosen to ensure that tags, where present, will always maintain the primary influence within the hybrid vector. Where no tags are

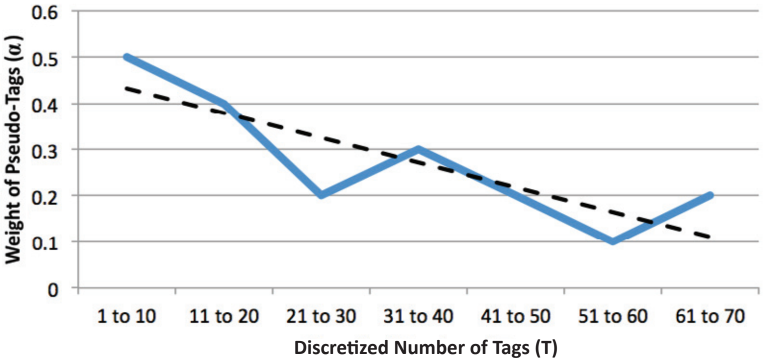


Figure 4: Weighting pseudo-tags against tags

present, the value of $c$ does not affect the hybrid vector. This is because the cosine similarity measure we use calculates the angle between normalised hybrid vectors, and so modifying $c$ may change pseudo-tag magnitudes, but will not change the angles computed.

Using this method of combining pseudo-tags with tags, we are now able to represent every track by a hybrid vector $h$. For tracks which are well-tagged, the tag vector will be the main contributor to $h$, and therefore allow recommendation to be made using the strengths of tag representation. For weakly-tagged tracks, any tags which they have will not be ignored, but instead enhanced by the inclusion of pseudo-tags in $h$. Recommendations are made using this hybrid representation based on cosine similarity, and will be referred to as **Hybrid**.

## 5. Evaluating Recommendation Quality

The objective of our evaluation is to investigate the performance of our `Hybrid` recommender system, compared with a purely tag-based recommender system, by evaluating the quality of recommendations made. To evaluate our approach to recommendation, we evaluate all the building blocks of the system. We first measure recommendation quality using only `PseudoTag`, and compare this to the quality of `Content` recommendations. This comparison allows us to quantify and understand the effect of our pseudo-tags when making recommendation. Next we compare the recommendation quality of our `Hybrid` model with the tag-based model. This evaluation allows us to understand the effects that pseudo-tags have on the system as a whole, and also the effect our `Hybrid` representation has across varying degrees of tag sparsity.

### *5.1. Experimental Design*

We structure our evaluation of recommendation quality as a query-by-example task, where each query is a single track. This type of recommender system is commonly found in systems where users can listen to single tracks, and then browse music based on the recommendations provided. We use 10-fold cross validation, and split the folds so that each track appears in the test set exactly once. For each query in our test set, we make 10 recommendations. To measure the quality of these recommendations, we use the association score measure developed in previous work [27]. This measure estimates the relationship between two tracks, based on the proportion of listeners who liked both tracks. Association score is calculated as

$$\text{association}(s_i, s_j) = \frac{\text{likes}(s_i, s_j)}{\text{listeners}(s_i, s_j)} \tag{16}$$

where $s_i$ and $s_j$ are the query and recommended tracks. $\text{listeners}(s_i, s_j)$ is the number of people who have listened to both track $s_i$ and track $s_j$, and $\text{likes}(s_i, s_j)$ is the number of people who have liked both $s_i$ and $s_j$.

Using Eq. (16) we are able to measure the quality of recommendations, on a scale of 0 to 1. A score of 0 indicates that there is no evidence of users liking a query-recommendation pair. A score of 1 indicates that the number of users who have liked both query and recommendation is equal to the number of estimated listeners of a query-recommendation pair, and the track is therefore a high quality recommendation. The scores we present in our results are the mean average association score achieved by all query-track pairs across all folds in our evaluation.

### *5.2. Datasets*

Our dataset contains 3174 tracks by 764 separate artists. The average number of tracks per artist in the collection is 4, and the most common artist in our collection has 78 tracks. The tracks fall into 12 distinct super-genres: Alternative (29%), Pop (25%), Rock (21%), R&B (11%); and Dance, Metal, Folk, Rap, Easy Listening, Country, Holiday and Classical making up the remaining 14% of the collection. We collect tag annotations for each track in our collection using the Last.fm Audioscrobbler API [1]. Last.fm provides normalised frequency values for the tags assigned to each track, with the most frequent tag always having a frequency of 100. Thus, these tag-frequencies are in the range 0 to 100 and there are 5160 unique tags in our dataset. We have access to the audio for this dataset, and so use the MFS representation to describe content. Pseudo-tags are learned from MFS neighbourhoods of size $K = 40$, based on empirical evaluation.

The evaluation data which we use to calculate the association score between two tracks was collected using the Last.fm Audioscrobbler API. We collect data for 175000 users over a period of 2 months, recording the tracks which a user gives a "thumbs up" through the Last.fm interface. On average, each user has liked, or given a "thumbs up" to 5.4 tracks in our collection. Further information that we collect using the Last.fm API is the number of listeners of each track in our collection, which we then use to estimate listeners$(s_i, s_j)$.

### *5.3. Recommendation Results*

We first evaluate the performance of our `PseudoTag` method against that of `Content` recommendation. Figure 5 shows the results obtained; the horizontal axis shows the number of recommendations that were made for each query, and the vertical axis shows the association score, measuring recommendation quality. The values presented are the mean average association score at the given number of recommendations. Significance at 95% is tested using a two-tailed paired Student's t-test, and error bars are shown also at a 95% confidence interval from the mean.

The `Content` recommender performance is illustrated by the dark dashed line, and `PseudoTag` by the dark solid line. When only 1 recommendation is made, both the `PseudoTag` and `Content` systems offer comparable performance. For 2 or more recommendations however, the `PseudoTag` representation is able to generate significantly better recommendations. The `PseudoTag` recommender system is also able to provide a more stable level of recommendation quality.

The improved quality and stability illustrate the power of using `PseudoTag` over `Content`. While the `PseudoTag` representation is based on content-based similarity, the tags learned do not necessarily describe content. This is the main advantage pseudo-tags have; additional knowledge from user tagging activities will also be discovered. Introducing this knowledge into the representation results in the improvements shown.

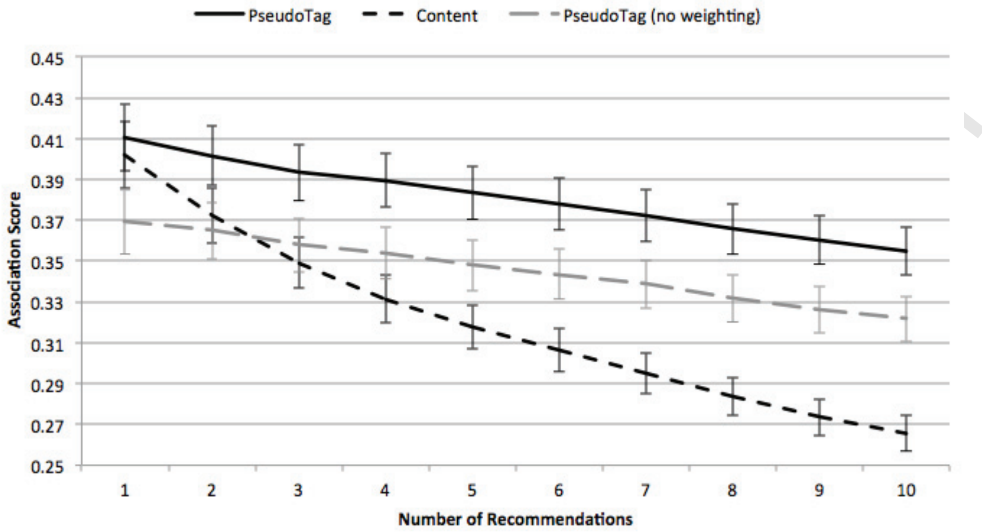


Figure 5: Recommendation Quality of Pseudo-Tags

The grey dashed line in Figure 5 shows recommendation quality for `PseudoTag` when the weighting scheme in Eq. (12) in not applied. This clearly shows that taking advantage of the order of tracks in content-based retrieval is advantageous, and not doing so reduces the quality of pseudo-tags.

Figure 6 shows our results comparing tag-based recommendation to our `Hybrid` approach. The format of this figure is the same as with Figure 5. The solid line shows recommendation quality for our `Hybrid` method, and the dashed line for tag-based recommendations.

For all numbers of recommendation, the `Hybrid` approach offers significantly better recommendations than the tag-based method. The strengthened representation offered by using pseudo-tags in our `Hybrid` system is able to influence recommendation in such a way that more relevant recommendations appear first in the recommendation list, resulting in this improved quality.

For tracks with a small number of tags, the pseudo-tag representation is given a higher weighting in the hybrid representation, and are allowed to control recommendation. It is important to remember that we inject a dynamic amount of pseudo-tags into our representation. This dynamic element is extremely helpful, since it allows tracks with a strong tag representation

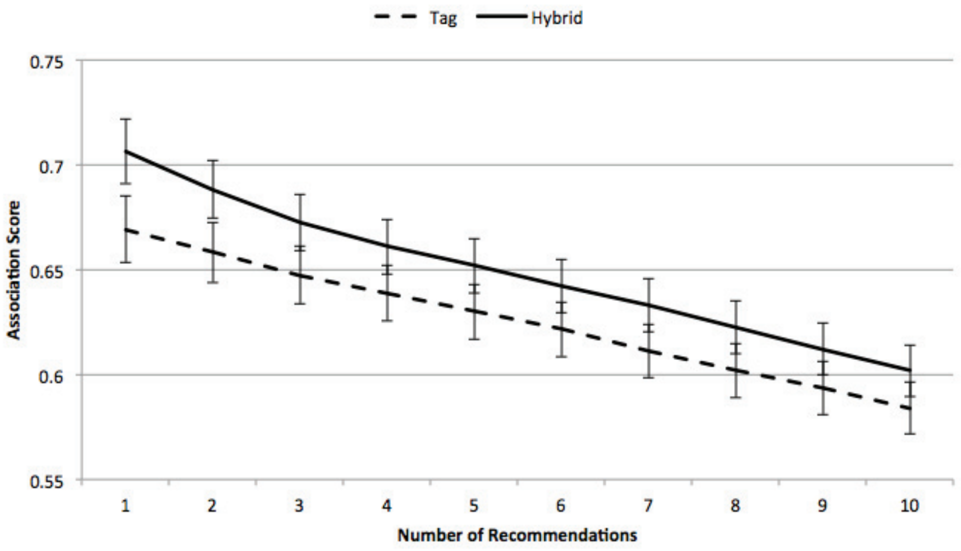


Figure 6: Recommendation Quality of Hybrid Representation

to still take advantage of this knowledge.

To further examine how this dynamic generation of a hybrid representation affects recommendation, we show the results by the number of tags a query has, in Figure 7. The vertical axis this time shows the difference in association score, between the Hybrid and tag-based approach.

The solid black line shows the improvement made when a query has between 0 and 9 tags, and the dashed black line shows the improvement made when a query has between 80 and 89 tags. These lines are the two extreme scenarios of cold-start and well-tagged in our system. To give a full picture, we also show the improvements made for between 10 and 79 tags as light grey lines, where each line shows a 10 tag interval.

As expected, the largest improvement made by using our hybrid approach is when a query has a very low number of tags. The improvement of 0.07 for 1 recommendation is greater than the improvement of the system as a whole. However, less expected is when the query has a large number of tags (80 to 89); our hybrid approach is still able to offer an improvement. We may only be adding a very small number of pseudo-tags to the tag-based representation, but they are meaningful and aid performance.

When we examine the set of recommendations made as a group, few

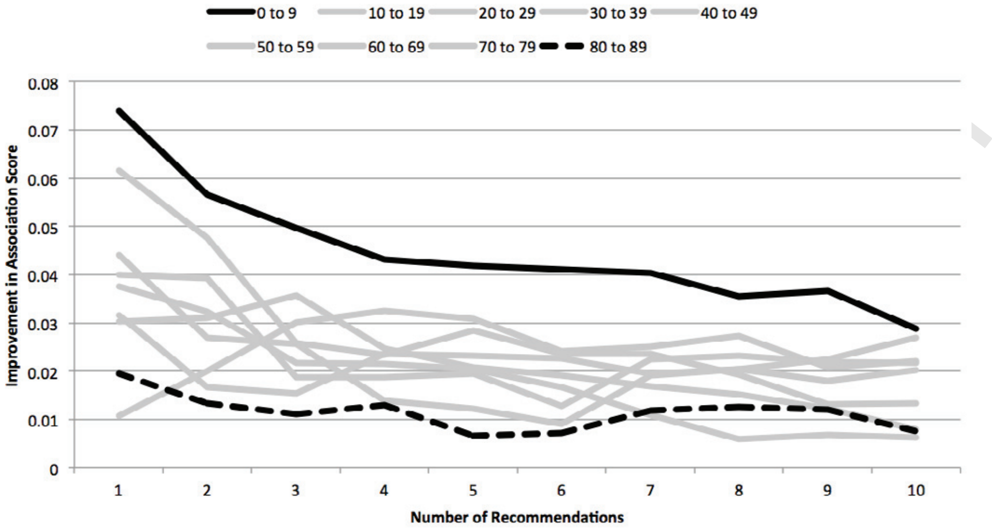


Figure 7: Improvement by Number of Tag Using Hybrid Representation

differences appear between the tag and hybrid groups. What does differ is the order in which these recommendations are presented. The effect of pseudo-tags on tracks with a large number of tags is refinement. The subtle differences introduced to the hybrid representation are enough to alter the specific ordering of recommendations in a positive way.

In our initial Figure 1 in Section 1 we define the challenge of cold-start, and argue that a desirable solution must provide recommendation quality within the grey shaded area. That is, the recommendation quality of a hybrid system must always be greater than the maximum quality of both content and tags, for all given numbers of tags in a representation. Figure 8 shows our results in the same format as Figure 1, and illustrates how each recommendation method performs as the number of tags a query has differs. The horizontal axis shows the number of tags the query track has, and the vertical axis shows the mean association score achieved for these tracks. The values shown are a moving average of the last 5 values, for example, the values shown for 20 tags is the average of the means for between 16 and 20 tags.

For our hybrid approach to meet the initial goal, its quality must always be greater than both that shown by `Content`, illustrated by the dashed grey line, and `Tag`, illustrated by the solid grey line. When the number of tags

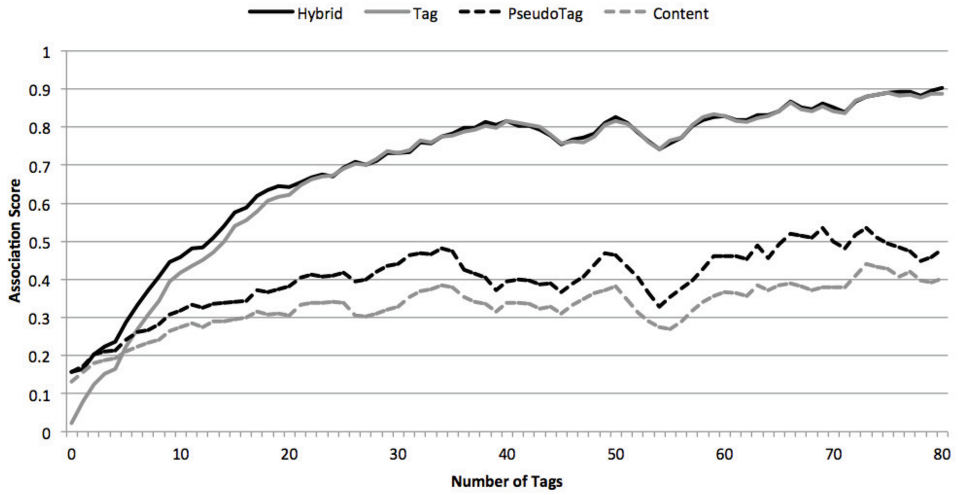


Figure 8: Quality at 5 Recommendations by Number of Tags

is less than 5, our hybrid approach must outperform Tag, and when the number of tags is 5 or more, our approach must outperform Content. The performance of Hybrid is shown by the solid black line, which for up to 20 tags clearly out-performs the Tag approach. For more than 20 tags Hybrid always performs at least as well as Tag (see Figure 7).

The reason that Hybrid is able to achieve this higher quality is because of the pseudo-tags that are embedded into the representation. These pseudo-tags allow the hybrid approach to have a distinct advantage when less tags are available, instead of reverting to the random recommendations of Tag. The quality of the PseudoTag method backing up the hybrid approach is illustrated by the dashed black line, and the quality of Content is illustrated by the dashed grey line. These results show that the ability to learn correct pseudo-tags can have a positive impact on recommendation quality regardless of how many tags a track has.

## 6. Evaluating Pseudo-Tagging Quality

To gain further insight into how our hybrid recommendation approach behaves, we examine the quality of pseudo-tags produced by our PseudoTag method. Pseudo-tags which are learned by our model should provide a precise representation for each track. If this is not the case, then our model may be learning pseudo-tags which introduce noise to a track's representation. Therefore, to evaluate pseudo-tags we compare our pseudo-tagging method with a similar current state-of-the-art auto-tagging method.

### *6.1. Dataset*

For this evaluation we use the CAL500 dataset [13]. The collection of 500 tracks is annotated by 174 tags, where each tag describes either instruments, vocals, genres, emotions, concepts or usage [13]. These tags were generated by 66 students, who collectively provided 1700 track-tag pairs. The creators of this dataset provide tag data in two formats: soft annotations which are weighted track-tag pairs, based on how the students voted on tags; and hard-annotations which identifies tags as relevant, if a predefined threshold is met.

The CAL500 dataset also comes with pre-extracted audio features, including chroma, dynamic-MFCC, and delta-MFCC feature vectors, and 32kbps 30-second clips of audio for each of the tracks. This combination of standard content and tag features provides an excellent small scale collection with which to evaluate MIR tasks, and has been used by many authors to evaluate auto-tagging [17, 36, 37]. However, instead of using the pre-extracted features to learn our pseudo-tags, we use our MFS representation, since this is what our `Hybrid` representation uses.

Figure 9 shows the full distribution of tag assignments throughout the CAL500 and our private collection described in Section 5.2. The horizontal axis shows the number of tags a track has, and the vertical shows the percentage of tracks which have this number of tags. The plot for the CAL500 dataset is created from the hard-annotations provided. As a result of the human annotation method used to collect tags, each track has between 13 and 43 tags, and there are no tracks with 0 tags. In contrast, the distribution observed for our private collection is very different. The general trend here is that the number of tracks decreases as the number of tags increases. This is to be expected in a real-world dataset, and is the result of a popularity bias within the collection [8]; popular tracks are more likely to be well-tagged. The average number of tags assigned to a track in our private collection is 34, with a standard deviation of 24.4. The more focused tagging in CAL500 make it more suited for auto-tagging experiments than our private collection. Evaluation of recommendation quality was not reported for CAL500, because its tagging in Figure 9 is very different from the private collection, which more accurately reflects what is observed in a popular online music service.

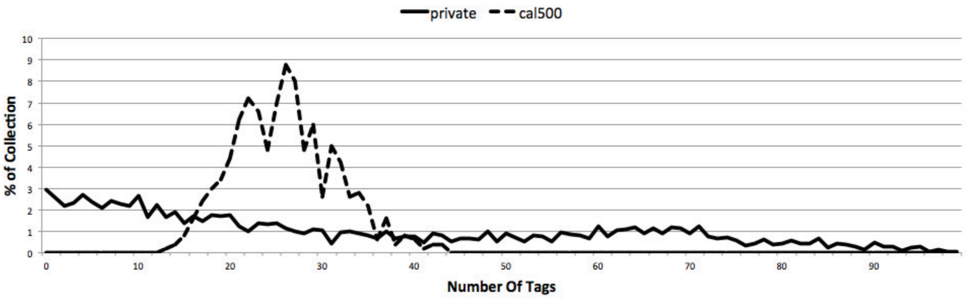


Figure 9: CAL500 and private dataset tag distributions

### 6.2. Experimental Design

The first stage of creating our hybrid representation is to generate pseudo-tags using content-based similarity, and so we aim to examine the quality of these pseudo-tags. Our method for generating pseudo-tags is based on summarising the tag vectors of the $K$ nearest neighbours of a track using Eq. (12). These neighbours are generated by a content-based recommender system using the MFS representation.

This task of generating pseudo-tags is very similar to auto-tagging, in that we are trying to generate the most suitable tags for a track. The main difference between standard auto-tagging and pseudo-tagging is that the evaluation criteria are slightly different. In auto-tagging, a wide range of evaluation measures are relevant [38] including precision, recall, f-score, accuracy and AUC-ROC, and the objective is to perform well in all measures.

Many evaluations of auto-tagging measure the per-tag precision and recall of the auto-tagger, that is, how well a fixed number of tracks can be assigned to a tag. In our case this evaluation does not make sense; all tags will be assigned to a track, but there is no guarantee that all tracks will be assigned a tag. We therefore measure the mean per-track precision and recall, which examines how well a fixed number of tags are assigned to a track. Precision measures the percentage of pseudo-tags correctly applied to a track, and recall is the percentage of a track's tags which are generated as pseudo-tags. Precision and recall are calculated as

$$\text{precision}(s) = \frac{\sum_{i=1}^{P} rel(p_i \| s)}{P} \tag{17}$$

$$\text{recall}(s) = \frac{\sum_{i=1}^{P} rel(p_i \| s)}{T} \tag{18}$$

where $rel(p_i\|s)$ is 1 if pseudo-tag $p_i$ has a weight greater than 0 in the tag vector of track $s$, and 0 otherwise. $T$ is the number of tags in the tag vector of track $s$, and $P$ is the number of pseudo-tags generated for track $s$.

For our pseudo-tags, we are most interested in achieving a high precision. We are not evaluating a classification task, or providing annotations for humans; we are trying to generate a representation to use in cold-start recommendation which correctly describes a track. Recall for the system will vary depending on how many pseudo-tags are required, but it is important that the precision for all levels of recall remains strong.

The average number of tags each track has in the CAL500 dataset is 26, and there are 174 possible tags in the collection. The mean precision of a random pseudo-tagger therefore is 0.15. However, our `PseudoTag` method is based on the $K$ nearest neighbours of a track. As a more meaningful baseline measure, we evaluate the performance of pseudo-tagging when tags are selected randomly from the $K$ nearest neighbours. This baseline will be referred to as **Random**.

We also compare our pseudo-tagging method with Sordo's current state-of-the-art $k$ nearest neighbour approach [23]. Sordo's method involves retrieving $N$ tracks for each track that must be annotated, and giving a higher influence to the tags of the nearest $K$ tracks. The influence of each track is calculated as

$$w(k) = \begin{cases} 1 & \text{if } k \leq K \\ \frac{1}{k^2} & \text{otherwise} \end{cases} \tag{19}$$

and these influence weights are then used to count the total influence of all tags of the nearest tracks, calculated as

$$p_i = \sum_{k=1}^{N} w(k) \lceil t_i(k) \rceil \tag{20}$$

where $\lceil t_i(k) \rceil$ evaluates to 1 if $t_i(k)$ is present, and 0 if it is not.

In Sordo's method, $N$ is set to the number of songs in the training set, and $K$ is set in the same way as for `PseudoTag`. The effect is that the nearest $K$ tracks all get an equal vote, and the votes of the remaining $N-K$ tracks have an inverse square decay.

### *6.3. Pseudo-Tagging Results*

The first measurement in our pseudo-tag evaluation compares how each method performs at varying levels of $K$. Figure 10 shows the mean precision obtained when 10 pseudo-tags are learned; the error bars shown are at a 95% confidence interval from the mean. The solid black line shows the precision obtained when using our `PseudoTag` method, the grey line shows that of Sordo's method, and the dotted line shows the baseline random method.

As expected, the performance of the random approach is much lower than both `PseudoTag` and Sordo. When $K$ is 10 both the `PseudoTag` and Sordo methods perform similarly, with a precision of approximately 0.68. As $K$ is increased however, each method starts to behave differently. Precision of our `PseudoTag` method improves until $K$ is around 40, whereas precision of the Sordo method gradually decreases. For values of $K$ greater than 10, our pseudo-tagging method achieves significantly better precision than Sordo.

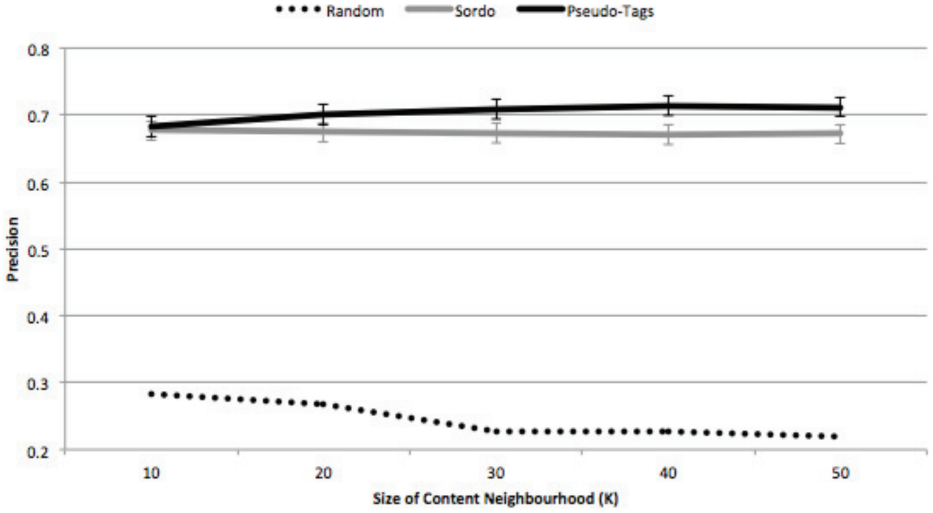


Figure 10: Precision learning 10 tags for different sizes of content neighbourhoods $K$

It is interesting that precision of the random method also decreases as $K$ is increased, with a large decrement occurring between $K = 20$ and $K = 30$. In our method we take the position of each track in the content retrieval list into account. In the Sordo and random methods, each track in the near neighbourhood is considered to be equally important, and thus the tag vectors of the $K$ nearest tracks are all considered equal. The reason that our `PseudoTag` method is able to achieve significantly increased precision over the Sordo method is that each of the $K$ nearest tracks are not treated equally. The weighting function that we apply allows the tags of the most similar tracks to have a larger influence, and to effectively control how tags are learned.

As $K$ is increased beyond 40, the precision of all methods decreases. This exact threshold is specific to the CAL500 dataset, but the trend is not. Increasing $K$ allows more unique tags to be found, and the weights of these tags will gradually increase as more tracks are included. With the random method, when $K$ is equal to the size of the training set, precision is reduced to that of the completely random system. With Sordo method, when $K$ is equal to the size of the training set, precision is reduced to what is achieved by assigning the most popular tags in the collection.

Figure 11 shows the recall-precision curve of each method, with a fixed $K = 40$ content neighbourhood. The points were generated by evaluating the precision and recall of each method for between 1 and 100 pseudo-tags. For both our `PseudoTag` and Sordo, precision decreases as the number of pseudo-tags learned is increased. This result is expected, since the chance of making an incorrect pseudo-tag assignment also increases. However, our pseudo-tag method that we use for recommendation, maintains a reasonable precision for all numbers of pseudo-tags. This is very important, and must be interpreted slightly differently to standard auto-tagging tasks.

When auto-tagging is used for suggesting tags to users, or providing a permanent tag representation, it is normal to suggest only 10 or 20 tags. For our task of creating a hybrid representation however, anywhere from 0 to 100 pseudo-tags may be required, depending on how well-tagged a track is. Regardless of how many pseudo-tags are required, we need them to be as precise as possible. It is therefore useful that our method is able to provide a competitive precision even when a large number of pseudo-tags is learned.

## 7. Conclusions

In this paper we have presented a method to improve music recommendation by tackling the problem of sparse tagging. We propose a dynamic hybrid representation that combines tags and pseudo-tags to reduce the sparsity of a track-tag representation matrix of a given dataset. There are two main contributions: the generation of pseudo-tags learned from similar tracks; and

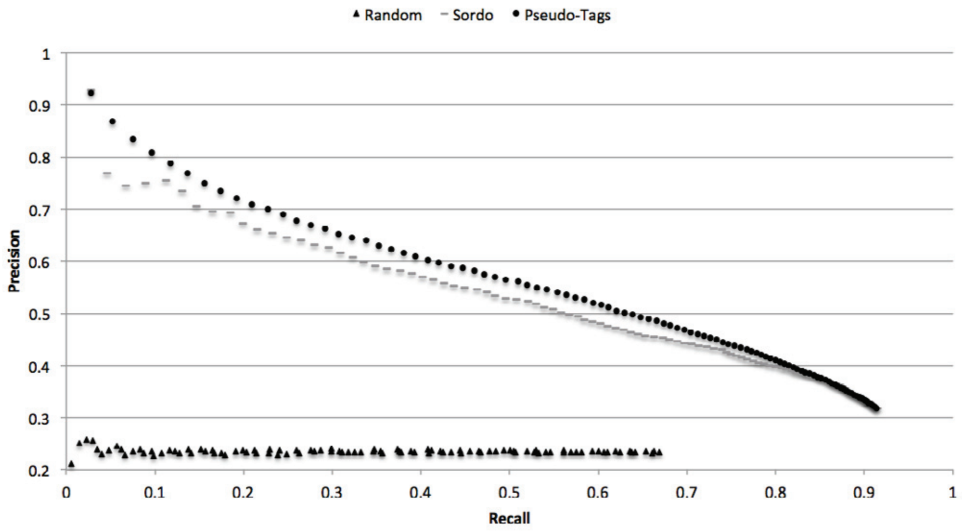


Figure 11: Recall Precision Curve

the dynamic combination of these pseudo-tags with existing tags.

Pseudo-tags are learned by aggregating tags from similar tracks; i.e. tracks whose audio content is similar according to the MFS music inspired texture representation. We note that pseudo-tags are not learned from content, they are learned from content and tags for other tracks. Tags capture human interpretations of low level music features as well as social/contextual concepts, and frequently used tags in musical neighbourhoods can provide a consensus of high level and social/contextual information. For tracks that are weakly tagged, this has the effect that pseudo-tags enable them to inherit human interpretation of content and social/contextual information, from the tags of similar tracks. In this way useful tag information is spread to neighbouring tracks in the content space.

The dynamic combination of these pseudo-tags with existing tags creates a hybrid representation which assigns more or less influence to pseudo-tags depending on how well-tagged a track is already. Sparsely tagged tracks attract more pseudo-tags, and these new tags are more equally weighted with existing tags; well-tagged tracks use existing tags to dominate the representation. Our Hybrid recommender system generates recommendations using these combined tags and pseudo-tags. Synergy from a hybrid system should be able to outperform its component parts, regardless of the sparseness of tagging. Our results demonstrate that Hybrid provides improved recommendations compared to Tag and Content individually. The benefit from the social and contextual information in this combined meta-data can be seen in good recommendations when tracks are sparsely tagged. There is also improved recommendation quality for well-tagged tracks, indicating that the small influence of pseudo-tags is not harmful but adds some diversification to the representation.

Sparsely tagged tracks suffer from the cold-start problem in music recommendation. Since learning pseudo-tags is based on content-based retrieval, pseudo-tagging does not suffer from cold-start, unless all similar tracks are untagged. For extreme cases of cold-start, when a track has no tags, pseudo-tags provide a stronger alternative to content. We have shown that pseudo-tags with high precision may be learned by taking advantage of the order of tracks in content-based retrieval.

In future work, it will be interesting to investigate a similar approach with different tasks/domains. For example, in a recommender system which uses collaborative filtering, it may be possible to reduce cold-start by learning pseudo-profiles for new users. In this case the hybrid representation would potentially consist of real user track plays, and pseudo-user track plays. There are other domains with dual representations that could benefit from using our dynamic approach to creating a hybrid representation; e.g. image retrieval with pixel content and incomplete captioning could exploit pseudo-captions; product recommendation with some available reviews could learn key comments for pseudo reviews.

[1] Last.fm, `http://www.last.fm/api` (Jun 2014).

[2] P. Lamere, Social tagging and music information retrieval, Journal of New Music Research 37 (2) (2008) pp. 101–114.

[3] K. Bischoff, C. S. Firan, W. Nejdl, R. Paiu, Can all tags be used for search?, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM), 2008, pp. 193–202.

[4] C. Kao-Li, T. Yang, W. Lee, Personalized multimedia recommendation with social tags and context awareness, in: Proceedings of the World Congress on Engineering, Newswood Limited, Vol. 2, 2011, pp. 6–8.

[5] M. Kaminskas, F. Ricci, Location-adapted music recommendation using tags, in: User Modeling, Adaption and Personalization, Springer, 2011, pp. 183–194.

[6] O. Celma, Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space, Springer-Verlag, 2010.

[7] C. Firan, W. Nejdl, R. Paiu, The benefit of using tag-based profiles, in: Proceedings of the Latin American Web Conference, 2007, pp. 32 –41.

[8] D. Turnbull, L. Barrington, G. Lanckriet, Five approaches to collecting tags for music, in: Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR), 2008, pp. 225–230.

[9] H. Halpin, V. Robu, H. Shepherd, The complex dynamics of collaborative tagging, in: Proceedings of the International World Wide Web Conference, 2007, pp. 211–220.

[10] Last.fm dataset, the official song tags and song similarity collection for the Million Song Dataset, http://labrosa.ee.columbia.edu/millionsong/lastfm (Jun 2014).

[11] T. Bertin-Mahieux, D. Ellis, B. Whitman, P. Lamere, The million song dataset, in: Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR), 2011, pp. 591–596

[12] O. Celma, P. Cano, From hits to niches?: How popular artists can bias music recommendation and discovery, in: Proceedings of the KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, ACM, 2008, pp. 1–8.

[13] D. Turnbull, L. Barrington, D. Torres, G. Lanckriet, Semantic annotation and retrieval of music and sound effects, IEEE Transactions on Audio, Speech, and Language Processing 16 (2) (2008) 467–476.

[14] L. Barrington, D. O'Malley, D. Turnbull, G. Lanckriet, User-centered design of a social game to tag music, in: Proceedings of the ACM SIGKDD Workshop on Human Computation, 2009, pp. 7–10.

[15] E. Law, L. Von Ahn, Input-agreement: A new mechanism for collecting data using human computation games, in: Proceedings of the 27th International Conference on Human Factors in Computing Systems, 2009, pp. 1197–1206.

[16] M. Mandel, D. Ellis, A web-based game for collecting music metadata, Journal of New Music Research 37 (2) (2008) 151–165.

[17] T. Bertin-Mahieux, D. Eck, M. Mandel, Automatic tagging of audio: The state-of-the-art, in: Machine Audition: Principles, Algorithms and Systems, IGI Publishing, 2011, pp. 334–352.

[18] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, M. Slaney, Content-based music information retrieval: Current directions and future challenges, Proceedings of the IEEE 96 (4) (2008) 668–696.

[19] L. Barrington, D. Turnbull, G. Lanckriet, Auto-tagging music content with semantic multinomials, in: Online extended abstracts of Music Information Retrieval Evaluation eXchange (MIREX), 2008, `www.music-ir.org/mirex/abstracts/2008/AT_barrington.pdf` (Oct 2014).

[20] D. Tardieu, C. Charbuillet, F. Cornu, G. Peeters, MIREX-2011 single-label and multi-label classification tasks: IRCAM-classification2011 submission, Online extended abstracts of Music Information Retrieval Evaluation eXchange (MIREX), 2011, `www.music-ir.org/mirex/abstracts/2011/TCCP2.pdf` (Sep 2014).

[21] J. Shen, W. Meng, S. Yan, H. Pang, X. Hua, Effective music tagging through advanced statistical modelling, in: Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2010, pp. 635–642.

[22] M. Sordo, C. Laurier, O. Celma, Annotating music collections: How content-based similarity helps to propagate labels, in: Proceedings of the 8th International Society for Music Information Retrieval Conference (ISMIR), 2007, pp. 531–534.

[23] M. Sordo, Semantic annotation of music collections: A computational approach, Ph.D. thesis, Universitat Pompeu Fabra (2012).

[24] J. Kim, B. Tomasik, D. Turnbull, Using artist similarity to propagate semantic information, in: Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR), 2009, pp. 375–380.

[25] B. McFee, G.R.G. Lanckriet, The natural language of playlists, in: Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR), 2011, pp 537–542.

[26] L. Wu, L. Yang, N. Yu, X-S. Hua, Learning to tag, in: Proceedings of the 18th International World Wide Web Conference, 2009, pp 361–370.

[27] B. Horsburgh, S. Craw, S. Massie, R. Boswell, Finding the hidden gems: Recommending untagged music, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), 2011, pp. 2256–2261.

[28] M. Levy, M. Sandler, Music information retrieval using social tags and audio, IEEE Transactions on Multimedia 11 (3) (2009) 383–395.

[29] K. Yoshii, M. Goto, Continuous PLSI and smoothing techniques for hybrid music recommendation, in: Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR), 2009, pp. 339–344.

[30] B. Hu, M. Guo, H. Zhang, A hybrid music recommendation system by M-LSA, in: the International Conference on Computational Intelligence and Natural Computing, 2009. Vol. 1, IEEE, 2009, pp. 129–132.

[31] A. Nanopoulos, D. Rafailidis, P. Symeonidis, Y. Manolopoulos, Musicbox: Personalized music recommendation based on cubic analysis of social tags, IEEE Transactions on Audio, Speech, and Language Processing, 18 (2) (2010) 407–412.

[32] B. Horsburgh, S. Craw, S. Massie, Music-inspired texture representation, in: Proceedings of the 26th AAAI Conference on Artificial Intelligence, 2012, pp. 52–58.

[33] P. Mermelstein, Distance measures for speech recognition, psychological and instrumental, Pattern Recognition and Artificial Intelligence 116 (1976) 91–103.

[34] S. Stevens, J. Volkmann, E. Newman, A scale for the measurement of the psychological magnitude pitch, Journal of the Acoustical Society of America 8 (1936) 185–190.

[35] T. Landauer, P. Foltz, D. Laham, An introduction to latent semantic analysis, Discourse Processes 25 (2-3) (1998) 259–284.

[36] M. Hoffman, D. Blei, P. Cook, Easy as CBA: A simple probabilistic model for tagging music, in: Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR), 2009, pp. 369–374.

[37] R. Miotto, L. Barrington, G. Lanckriet, Improving auto-tagging by modeling semantic co-occurrences, in: Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR), 2010, pp. 297–302.

[38] MIREX (Music Information Retrieval Evaluation eXchange), `http://music-ir.org/mirex/wiki` (Nov 2014).

[39] F. Font, J. Serra, X. Serra, Folksonomy-based tag recommendation for online audio clip sharing, in: Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR), 2012, pp. 73–78.

[40] F. Font, J. Serra, X. Serra, Folksonomy-based tag recommendation for collaborative tagging systems, International Journal on Semantic Web and Information Systems 9 (2) (2013) 1–30.