# Comprehensive Query-Dependent Fusion using Regression-on-Folksonomies: A Case Study of Multimodal Music Search

Bingjun Zhang[1], Qiaoliang Xiang[1], Huanhuan Lu[1], Jialie Shen[2], Ye Wang[1]
[1]School of Computing, National University of Singapore
[2]School of Information Systems, Singapore Management University
[1]{bingjun,xiangqiaoliang,luhuan,wangye}@comp.nus.edu.sg, [2]jlshen@smu.edu.sg

## ABSTRACT

The combination of heterogeneous knowledge sources has been widely regarded as an effective approach to boost retrieval accuracy in many information retrieval domains. While various technologies have been recently developed for information retrieval, multimodal music search has not kept pace with the enormous growth of data on the Internet. In this paper, we study the problem of integrating multiple online information sources to conduct effective query dependent fusion (QDF) of multiple search experts for music retrieval. We have developed a novel framework to construct a knowledge space of users' information need from online folksonomy data. With this innovation, a large number of comprehensive queries can be automatically constructed to train a better generalized QDF system against unseen user queries. In addition, our framework models QDF problem by regression of the optimal combination strategy on a query. Distinguished from the previous approaches, the regression model of QDF (RQDF) offers superior modeling capability with less constraints and more efficient computation. To validate our approach, a large scale test collection has been collected from different online sources, such as Last.fm, Wikipedia, and YouTube. All test data will be released to the public for better research synergy in multimodal music search. Our performance study indicates that the accuracy, efficiency, and robustness of the multimodal music search can be improved significantly by the proposed folksonomy-RQDF approach. In addition, since no human involvement is required to collect training examples, our approach offers great feasibility and practicality in system development.

## Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation, Search process; H.5.5 [**Sound and Music Computing**]: Systems

## General Terms

Algorithms, Design, Experimentation


*MM'09,* October 19–24, 2009, Beijing, China.


## Keywords

Music, multimodal search, query-dependent fusion, folksonomy

## 1. INTRODUCTION

As the amount of information on the Internet grows dramatically, the information retrieval (IR) technology advances by combining multiple complementary sources to better satisfy users' information need. Particularly, multimodal based scheme is becoming one of the most important trends in media (such as audio, image and video) retrieval today. It holds great potential to be applied in many different applications including: *web search* where the relevance of a web page to a text query is scored based on its body text, anchor text, and its linking relation to other pages [8]; *multimedia search* where the textual metadata of a video, a image, or a music track (e.g., titles, tags, descriptions) and their content features (e.g., motion intensity, texture, timbre) are combined to rank the media documents [27, 16, 4]; and *meta-search* where multiple ranked lists from different search engines are fused into a more relevant one [21]. The development of advanced fusion techniques enables IR to fully unleash the power of information from different modalities. Leveraging this technology will allow retrieval systems to deliver better quality results over a wide range of queries.

Currently, the most simple and efficient fusion approach is Query Independent Fusion (QIF), which applies the same combination strategy in search to any type of queries. The disadvantage of this approach is its low query accuracy and poor scalability to cover the high complexity of various query topics. Research shows that different underlying search modalities could provide different levels of contribution to the final performance of a retrieval process. This observation suggests the superiority of Query Dependent Fusion (QDF) strategies, where the combination strategy of a search can be adjusted based on user's search intention. The desirable solution of QDF is to derive the optimal combination strategy for each possible query of a targeted application. While QDF achieves better performance against QIF, existing approaches still suffer from two main weaknesses.

Firstly, previous QDF approaches [3, 27, 10, 26, 25, 9] relied on manually designed queries in the TRECVID test collections [23] as the training samples. Due to the high labor cost in designing query topics, only a small number of queries ($\sim$100) were available for training and evaluating a system which is to be used against a large number of unseen

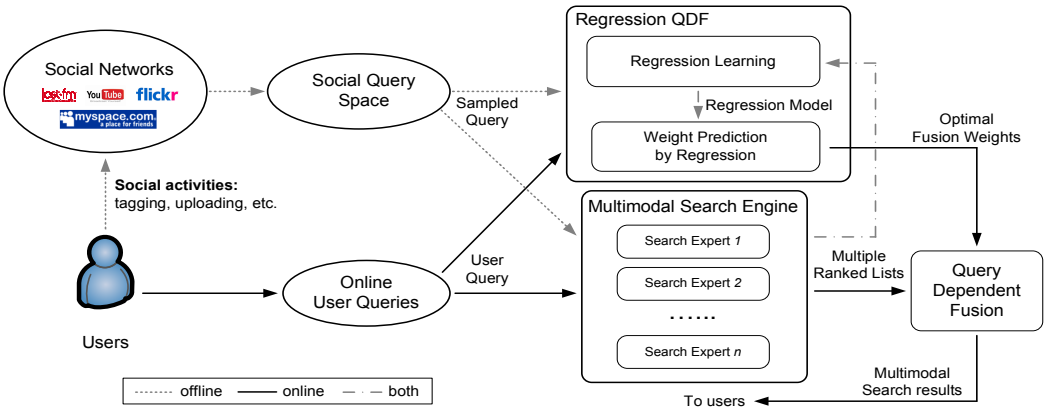



**Figure 1:** The framework of regression-on-folksonomy based query-dependent fusion for effective multimodal search.

user queries. This approach is less practical for two reasons: 1) it increases labor cost greatly in designing queries during the system development; 2) it also significantly limits the knowledge scope of the training queries, which may result in poor generalization performance when predicting combination strategies on unseen user queries.

Secondly, many existing QDF schemes [3, 27, 10, 26, 25, 9] predicted the combination strategy for each query by following the procedure of query feature extraction, query matching, and combination weight optimization. The query matching step may seem redundant and less efficient, if it is possible to learn a model which directly predicts the optimal combination strategy from certain features of each query. The direct prediction of combination strategy may also lessen the model constraints imposed by the explicit query matching. In addition, it would be important to achieve an efficient fusion and online learning process to accommodate and learn from a large number of unseen user queries simultaneously. Unfortunately, no existing study investigates how to improve the efficiency and the adaptive learning aspect of QDF for such purposes.

Motivated by the above observations, this paper studies the problem of how to effectively combine heterogeneous knowledge sources to facilitate accurate and efficient QDF for multimodal music search. We design and implement a novel framework called "folksonomy-RQDF" whose architecture is illustrated in Fig. 1. Distinguished from the previous QDF approaches only considering very limited amount of knowledge, our approach contributes to QDF research for multimodal music search in the following aspects.

**A comprehensive knowledge base extracted from the World Wide Web** - With the phenomenal growth of social network sites, such as Last.fm, MySpace, YouTube, and Wikipedia,[1] researchers have noticed that search queries and online tags created by a common user population are strongly coupled in expressing users' information need [13]. Researchers also explored the possibility to construct an ontology of the knowledge space of users' information need from folksonomy data [14]. In this work, we follow those beliefs and explore the possibility of automatically forming user queries based on the online folksonomy data. Those automatically-generated queries can be used as comprehensive training and evaluation data in the QDF system.

[1] http://www.last.fm, http://www.myspace.com, http://www.youtube.com, http://www.wikipedia.com,

**Automatic training example generation** - This approach saves a large amount of human labor in forming training examples. It enables the training of the QDF system on a comparable knowledge scope to the potential unseen queries, which will significantly improve the generalization performance of the system against unseen queries. As shown in our music search experiments in Section 6, the retrieval accuracy can be boosted by a statistically significant amount when more automatically-generated queries are used for training.

**Advanced regression model for effective QDF** - In this study, the QDF process is modeled as a regression problem from a query to its combination strategy. This concept not only removes the model constraints of the query matching step for better effectiveness, but also eliminates the redundant computation during the online prediction of the combination strategy. It also simplifies the query feature extraction by employing a concise and efficient document vector [12] model instead of using complex natural language processing (NLP) techniques in the previous QDF methods. In addition, we propose an efficient algorithm to learn a Support Vector Regression model, whose run-time has inverse dependence on the size of the training set.

A comprehensive experimental study shows that the proposed framework enjoys superior retrieval accuracy, efficiency (in both offline training and online prediction), and robustness in the multimodal music search application compared with the previous QDF methods. Consequently, those advantages enable our approach to offer better practicality for the system development in many real-life applications.

The paper is organized as follows. Section 2 reviews the research development of multimodal fusion in the IR field. Section 3 describes the construction of a social query space from online folksonomy data and the automatic formation of social queries . Section 4 formulates the proposed regression model for QDF. Section 5 and 6 detail the experimental setup and results followed by our conclusion in Section 7.

## 2. RELATED WORK

Multimodal fusion is an important research problem in information retrieval and multimedia systems. Existing techniques can be categorized into: query-independent fusion (QIF) and query-dependent fusion (QDF) schemes. In this section, we review the two schemes and their advantages.

QIF approaches apply the same combination strategy of multiple search experts to all queries. It assumes that var-

**Table 1:** The comparison of different fusion schemes for multimodal search.

| Scheme | Query Formation | Query Feature | Query Matching | | Weight Optimization | Tested App. |
|---|---|---|---|---|---|---|
| | | | Model | Procedure | | |
| QIF | Manually design a small number ($\sim$ 100) of queries. | N.A. | N.A. | N.A. | N.A. | Text/video search, etc. |
| CQDF-Single | | Semantic concepts extracted from text by NLP techniques. Features extracted from media (video or image). | Designed or auto-clustered classes. | Classify a query to a single class. | Compute the weights of the best MAP on the matched queries. | Video search. |
| CQDF-Mixture | | | Auto-discovered classes by pLQA. | Match a query to a mixture of classes. | | Video/meta-search. |
| QDF-KNN | | | The raw training queries. | Match a query to the first $K$ nearest neighbors. | | Video search. |
| Folksonomy-RQDF | Automatically form millions of queries. | The document vector [12] of text. Features extracted from media (audio). | A regression model. | N.A. | Directly predict the weights. | Music search. |

ious modalities enjoy a fixed contribution to the retrieval performance regardless of the actual query topics. One typical QIF method was proposed by Shaw and Fox for text retrieval [21]. The main advantage of QIF methods is their computational efficiency and simplicity. However, it does not provide adaptive fusion solutions to varied query topics of users' information need. QIF methods suffer from the fact that the performance of an individual modality varies considerably for different query topics.

In this case, QDF becomes a natural solution. It offers better adaptiveness for various query types. In the methods [27, 3], the training queries were manually designed by domain experts. A limited number of query-classes were manually discovered based on the query topics with the hope that all queries in a class share similar combination weights. This approach suffers from two main disadvantages. Firstly, it is highly complex to determine whether the actual underlying combination weights of the queries in each class are similar. In addition, domain knowledge and human efforts are needed to define meaningful classes. In [10, 9], a clustering approach was proposed to automatically discover classes based on the manually designed query pool of TRECVID [23]. All queries in a query class share more similar combination weights compared to the approaches with manually discovered classes. However, a common combination strategy is used for all user queries that are classified into a class regardless of the query topic and combination-weight difference within a class. These class-based query dependent fusion approaches with a single class to represent user queries are termed "CQDF-Single" in this paper. To achieve better fusion effectiveness, Yan et al. proposed the probabilistic latent query analysis (pLQA) [26]. The key innovation is that combination weights of an incoming query can be reconstructed by a mixture of query classes (termed "CQDF-Mixture"). The scheme has been evaluated in video retrieval over TRECVID'02$\sim$'05 collections and meta-search on the TREC-8 collection. This approach offers better resolution in a query-to-combination-weights mapping. However, its estimation model assumes that different queries in each query class share the same combination weights. The latest QDF method proposed by Xie et al. [25] represented a user query by the linear combination of its first $K$ nearest neighbors in the raw training query set (termed "QDF-KNN"). This QDF model offers better resolution from query-to-combination-weights mapping, but suffers from high computational load of nearest neighbor searching in a large training set.

A detailed comparison between previous multimodal fusion methods and the proposed folksonomy-RQDF approach is summarized in Table 1. We can see that the previous QDF methods impose expensive human involvement in query or class design which greatly limits their feasibility in real-life applications. In addition, the previous QDF schemes are mainly developed for text or video retrieval. To the best of our knowledge, no scheme has been proposed or tested for large-scale music information retrieval.

# 3. AUTOMATIC QUERY FORMATION

Due to human labor constraints, the previous QDF approaches [3, 27, 10, 26, 25, 9] relied on a small number ($\sim$100) of manually designed queries to train a query-to-combination-weights mapping. The limited knowledge scope of users' information need represented by the few handcrafted queries greatly diminishes the generalization performance of QDF systems against a large number of unseen user queries issued from a much wider knowledge scope of users' information need.

To overcome this problem, we propose a method to automatically form a large number (in millions) of queries from readily available online folksonomy data. In order to simulate unseen user queries in a comprehensive manner, the automatically-formed queries from folksonomy data (termed "social queries") should have the following properties:

**P.1**: Social queries cover a comparable knowledge scope of the users' information need in unseen user queries;

**P.2**: They have the same semantic structure as unseen user queries;

**P.3**: They reveal similar distribution of unseen user queries over various query topics.

In this section, we describe the automatic formation of social queries from folksonomy data with a case study in multimodal music search domain. Validated in our experiments, the formed social queries fulfilled the above properties and contributed to the superior generalization performance of the QDF system.

## 3.1 Folksonomies to Social Query Space

In [13], the authors suggested that searching and tagging are dual activities governed by the information need of a common user population, and it is reasonable to use tag data to approximate user queries in analyzing user behaviors and improving search accuracy. Therefore, by using the online tag records (folksonomy) as the vocabulary in forming social queries, P.1 of the formed queries will be preserved.

Based on the tag set of the folksonomy, the simplest way to automatically form social queries is to randomly select a

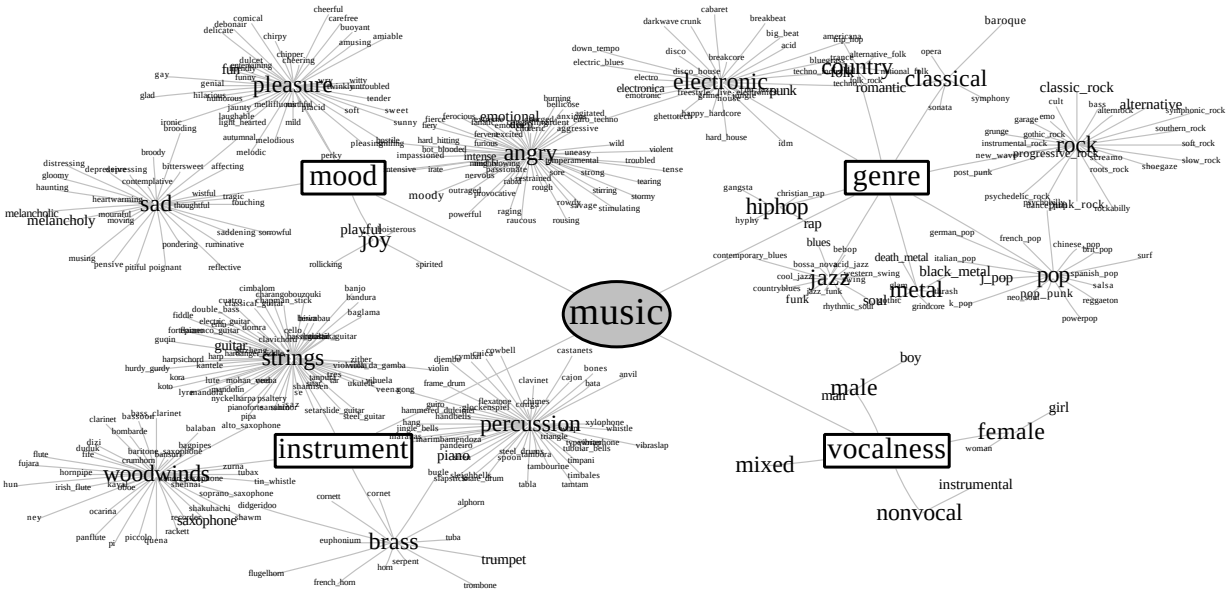



**Figure 2:** The semantic structure of the music social query space. The font size of a tag indicates its popularity on Last.fm.

**Table 2:** The contribution of each online resource in constructing the music social query space.

| Resource | Contribution | Details |
|---|---|---|
| Last.fm, MySpace, Wikipedia | Raw set of concepts/tags | 244 tags were collected for the genre dimension; 286 tags for the mood dimension; 454 tags (instrument names) for the instrument dimension; and 13 tags for the vocalness dimension. |
| Google | Spell checks | N.A. |
| Wikipedia, WordNet | Semantic source for homonyms, synonyms and hyponyms | Wikipedia disambiguation pages were used for homonyms. Particular Wikipedia pages, like "Genre#Music_genres", "List_of_musical_instruments", were used to find hyponyms. WordNet is used to find synonyms and hyponyms. All labels were grouped into clusters: 8 clusters for genre, 4 clusters for mood, 4 clusters for instrument, and 4 clusters for vocalness. |

reasonable number of tags as a description of a user's information need and concatenate the tags together as a social query. However, a set of plain tags does not suffice to generate meaningful user queries. Semantic structures or relations among the tags are required to form semantically consistent queries. For example, the query "Find me a happy and sad song" would not be formed, if the antonym relationship is established between the plain tags "happy" and "sad".

Therefore, to form consistent queries, a semantic network (ontology) needs to be established on top of the folksonomy data in order to put logically founded constraints on the use of the plain tags. Based on the modeling approach in [14], a lightweight ontology can be constructed from the folksonomy data by folding the Actor-Concept-Instance (user-tag-music item) model into a semantic network emerged from either overlapping communities or overlapping sets of instances. According to the integrated approach to turning folksonomies into ontologies in [5], the lightweight ontology can be further enriched by numerous online resources: 1) lexical resources like dictionaries and Google[2]; 2) ontologies and semantic web resources, such as WordNet [15] and Wikipedia. With the folksonomy data as the tag base and the constructed ontology as the semantic structure, semantically consistent queries can be automatically formed, which preserves P.2 of the formed social queries.

To further simulate the distribution of user queries over different query topics, we introduce an additional property for each tag, i.e., popularity. The popularity of a tag can be implemented as the number of times it has been used by an actor to tag an instance. Tag popularity indicates the frequency that the information represented by the tag is requested by users, which in turn indicates the inherent distribution of a query topic. The social query space augmented with tag popularities to represent the query topic distribution preserves P.3 of the formed social queries.

Based on the above discussions, a social query space for content-based multimodal music search was constructed using the folksonomy data of Last.fm, MySpace, and Wikipedia plus the synonym and hyponym relations in WordNet. The detailed contribution of each online resource in constructing the social query space is listed in Table 2. A pictorial representation of the social query space is shown in Fig. 2. The tree structure represents the semantic relations between tags, which serves as the logical constraints in forming consistent social queries. As can be seen from the music social query space, music content can be described from multiple music dimensions, e.g., genre, mood, instrument, and vocalness. In each dimension, people use different concepts/tags to describe the music styles. The concepts in each music dimension are further grouped into meaningful clusters.

## 3.2 Social Query Sampling

From the social query space, we need to further specify a set of rules to sample consistent social queries in simulating unseen user queries. The rules for the music social query space are: 1) the query cannot be empty (at least one tag

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

should be sampled); 2) the number of sampled tags should be less than the number of keywords in a reasonable query issued by human (set as 5 in this case study); and 3) no conflicting tags (tags from different clusters of the same music dimension) should be sampled to form a single query.

Based on the above rules and the semantic structure of the social query space, tags were sampled based on their distribution to form a large number of social queries. Some exemplar raw queries are "classical violin", "happy female country", etc. By augmenting the raw queries with "Find me the music with style" heading, social queries can be formed to simulate unseen user queries in searching music documents with certain content styles. From the music social query space, more than 448 million unique social queries could be formed to cover a wide scope of unseen users' information need in searching music tracks by their content styles. These comprehensive social queries are used to train and evaluate a better generalized the QDF system discussed in the following sections.

# 4. REGRESSION MODEL FOR QDF

In this section, we propose a novel approach to model the QDF problem by regression. In addition, we extend the efficient algorithm Pegasos [22] for learning binary-class Support Vector Machines (SVM) to its online regression version (ORPegasos) for learning Support Vector Regression (SVR) models. We also prove that ORPegasos enjoys same efficiency property as the original Pegasos: its run-time has inverse dependence on the training set size in learning a regression model with the same generalization performance.

## 4.1 Model Definition

A textual user query can be modeled as a real vector $\mathbf{q}$ by the document vector model (DV) [12] where an element 1 indicates the presence of a vocabulary word in the query while an element 0 indicates its absence.[3] The dimension of $\mathbf{q}$ equals to the number of words in the vocabulary of all possible queries.[4] The combination strategy of the underlying multiple search experts can be formulated as a weight vector $\mathbf{W} = [W_1, ..., W_N]^T$, where $N$ is the number of search experts, $0 \leq W_j \leq 1$, and $\sum W_j = 1$. The QDF problem which derives the optimal $\mathbf{W}$ for each $\mathbf{q}$ can be modeled as multiple regression of $\mathbf{W}$ on $\mathbf{q}$: $\mathbf{W} = f(\mathbf{q})$. The multiple regression model can be further decomposed into a series of single regression models of $W_j$ on $\mathbf{q}$ as: $W_j = f_j(\mathbf{q})$ [7]. After normalization of $W_j$, each of the single regression models can be used for deriving the combination weight of an underlying search expert.

To realize a single regression model, we employ the Support Vector Regression (SVR) formulation [20], $y = \langle \mathbf{w}, \mathbf{x} \rangle + b$, due to its superior generalization performance and simple modeling approach. For clearer notation, we use $\mathbf{x}$ and $y$ to represent the query $\mathbf{q}$ and its combination weight $W_j$ of a single search expert respectively in the following text. Based on a training set of query-weight pairs $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ (Section 5.3 details the generation of the training data), SVR derives its model $\mathbf{w}$ by minimizing the unconstrained empirical loss function plus a regularization term of $\mathbf{w}$:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_{(\mathbf{x},y)\in\mathcal{S}} \ell(\mathbf{w}; (\mathbf{x}, y)), \quad (1)$$

where

$$\ell(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, |y - \langle \mathbf{w}, \mathbf{x} \rangle| - \epsilon\} \quad (2)$$

is the $\epsilon$-insensitive empirical loss function. The bias term, $b$, in the original SVR problem is implicitly incorporated in $\mathbf{w}$, which amounts to adding one more constant feature dimension with value 1 in each sample $\mathbf{x}$.

In the following sections, we first derive a batch regression version of Pegasos (RPegasos) for learning SVR models based on an initial training set of query data. Then, we extend RPegasos into its online version ORPegasos for online learning based on unseen queries.

## 4.2 Regression Pegasos

Based on the ideas of Pegasos [22] for binary-class SVM, the proposed RPegasos algorithm solves the SVR optimization problem of Eq. (1) in a stochastic sub-gradient manner. RPegasos performs $T$ iterations with $k$ samples randomly chosen to calculate the sub-gradient at each iteration. Initially, $\mathbf{w}_1$ is set to the zero vector. On iteration $t$, we form a set $\mathcal{A}_t \subseteq \mathcal{S}$ of size $k$ and Eq. (1) can be approximated by

$$f(\mathbf{w}; \mathcal{A}_t) = \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{k}\sum_{(\mathbf{x},y)\in\mathcal{A}_t} \ell(\mathbf{w}; (\mathbf{x}, y)). \quad (3)$$

Next, $\mathbf{w}_t$ can be updated in two steps. The first is the sub-gradient step formulated as:

$$\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \eta_t \bigtriangledown_t, \quad (4)$$

where

$$\bigtriangledown_t = \lambda \mathbf{w}_t - \frac{1}{|\mathcal{A}_t|}\sum_{(\mathbf{x},y)\in\mathcal{A}_t^+} \mathbf{sign}(y - \langle \mathbf{w}_t, \mathbf{x} \rangle)\mathbf{x} \quad (5)$$

is the sub-gradient of $f(\mathbf{w}_t; \mathcal{A}_t)$ at $\mathbf{w}_t$ and $\eta_t = 1/(\lambda t)$ is the learning rate. $\mathcal{A}_t^+$ is formed by $(\mathbf{x}, y) \in \mathcal{A}_t$ but with non-zero loss. The second step is the projection of $\mathbf{w}_{t+\frac{1}{2}}$ onto the set $\mathcal{B} = \{\mathbf{w} : ||\mathbf{w}|| \leq 1/\sqrt{\lambda}\}$, in which the optimal $\mathbf{w}$ resides. Therefore,

$$\mathbf{w}_{t+1} = \min\{1, 1/(\sqrt{\lambda}\ ||\mathbf{w}_{t+\frac{1}{2}}||)\}\ \mathbf{w}_{t+\frac{1}{2}}. \quad (6)$$

Based on the representer theorem [20], the kernel version of RPegasos can be obtained by representing $\mathbf{w}_t = \sum_{i\in\mathcal{I}_t} \alpha_i \mathbf{x}_i$, where $\mathcal{I}_t$ is a subset of $\{1, ..., m\}$, and calculating $\langle \mathbf{w}_t, \mathbf{x}_t \rangle = \sum_{i\in\mathcal{I}_t} \alpha_i \langle \mathbf{x}_i, \mathbf{x}_t \rangle$ and $||\mathbf{w}_t||^2 = \sum_{i,j\in\mathcal{I}_t} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$.

## 4.3 Online Regression Pegasos

Now we can extend RPegasos into its online version ORPegasos. The SVR will be dynamically adjusted according to future user queries when predicting the optimal combination weights. Similar to the batch version, the ORPegasos performs sub-gradient update and projection. However, the main difference is the instantaneous loss function at the $i$-th online sample $(\mathbf{x}_i, y_i)$, which can be defined by

$$f(\mathbf{w}; \mathbf{x}_i, y_i) = \frac{\lambda}{2}||\mathbf{w}||^2 + \ell(\mathbf{w}; (\mathbf{x}_i, y_i)). \quad (7)$$

The sub-gradient update rule is derived as below,

$$\mathbf{w}_{i-\frac{1}{2}} = (1 - \eta_t\lambda)\mathbf{w}_{i-1} + \eta_t \mathbf{sign}(y_i - \langle \mathbf{w}_{i-1}, \mathbf{x}_i \rangle)\mathbf{x}_i, \quad (8)$$

[3] Bold letters notate column vectors. Italic letters (lower and upper cases) notate scalars. Calligraphic upper case letters notate sets.

[4] As noted, $\mathbf{q}$ is inevitably high dimensional and sparse (only a few words in a query generate 1s in its DV). In practice, this does not degrade the computational efficiency of our model. On the contrary, its sparsity enables better generalization performance in a linear SVR model, thus provides better efficiency by avoiding kernels.

**Algorithm 1**: The algorithm of ORPegasos.

**Input:** Offline initial training set, $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$;
Online individual training pair, $(\mathbf{x}_{m+i}, y_{m+i})$;
**Output:** Learned SVR model, $\mathbf{w}$;
**Description:**
**Offline:**
Initialize $\mathbf{w}_1$ as the zero vector;
**for** $t = 1$ to $T$ **do**
  Randomly choose $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = k$;
  Set $\mathcal{A}_t^+ = \{(\mathbf{x}, y) \in \mathcal{S}) : |y - \langle \mathbf{w}_t, \mathbf{x} \rangle| < \epsilon \}$;
  Set $\eta_t = 1/(\lambda t)$;
  Set $\mathbf{w}_{t+\frac{1}{2}} =$
    $(1 - \eta_t \lambda)\mathbf{w}_t + \frac{\eta_t}{k} \sum_{(\mathbf{x},y) \in \mathcal{A}_t^+} \mathbf{sign}(y - \langle \mathbf{w}_t, \mathbf{x} \rangle)\mathbf{x}$;
  Set $\mathbf{w}_{t+1} = \min\{1, 1/(\sqrt{\lambda}\ ||\mathbf{w}_{t+\frac{1}{2}}||)\}\ \mathbf{w}_{t+\frac{1}{2}}$;
**end for**
**return** $\mathbf{w}_T$ as the offline model $\mathbf{w}_m$.
**Online:**
Set $\mathbf{w}_{m+i-\frac{1}{2}} =$
  $(1 - \eta_t \lambda)\mathbf{w}_{m+i-1} + \eta_t \mathbf{sign}(y_{m+i} - \langle \mathbf{w}_{m+i-1}, \mathbf{x}_{m+i} \rangle)\mathbf{x}_{m+i}$;
Set $\mathbf{w}_{m+i} = \min\{1, 1/(\sqrt{\lambda}\ ||\mathbf{w}_{m+i-\frac{1}{2}}||)\}\ \mathbf{w}_{m+i-\frac{1}{2}}$;
**return** $\mathbf{w}_{m+i}$ as the online model.

and the projection step of $\mathbf{w}_{i-\frac{1}{2}}$ to $\mathbf{w}_i$ is the same as Eq. (6). Following the work in [11], ORPegasos can be easily extended into its kernel version.

The pseudo-code of RPegasos and ORPegasos is shown in Alg. 1, which takes an offline initial training set and an online individual training pair (an user query plus its oracle weight vector discussed in Section 5.3) as the input data to simulate the real-life requirement of QDF systems.

Based on the analysis in [19, 22] with the binary-classification loss term replaced by the regression loss term, we can easily prove that ORPegasos converges in $\tilde{O}(\frac{1}{\lambda \delta \xi})$ iterations for learning a $\xi$ optimal SVR with confidence $1 - \delta$. Less number of iterations are required to achieve the same generalization performance, when more training samples are available [22]. The proof details are omitted due to space constraints. Therefore, the run-time of ORPegasos has the same inverse dependence as Pegasos, i.e., the more training samples available, the faster ORPegasos can learn a SVR with the same generalization performance. In addition, the number of support vectors in the kernel version of ORPegasos does not depend on the number of training samples, which is a desirable property for fast online SVR prediction. In summary, ORPegasos learns the same well generalized SVR faster in larger databases while keeping the same model complexity for fast online prediction.

## 4.4 Class-based v.s. Regression-based QDF

Compared with class-based QDF (CQDF) approaches, the proposed regression-based QDF (RQDF) directly predicts the optimal combination weights for each online user query without any query matching computation. RQDF enjoys several advantages. First, it is much faster to predict the optimal combination weights as it only needs to perform one inner product in linear SVR or a small number of kernel evaluations in its kernel version. In contrast, CQDF needs to conduct extra expensive query classification [3, 27, 10], calculation of the query-class mixture [26], or nearest query searching [25]. Second, RQDF adapts better to future user queries than CQDF, because the extra underlying class structure of CQDF makes it hard to adapt and update based on online queries. Third, it simplifies the query feature extraction by using an efficient DV model instead of complex natural language processing (NLP) techniques [3, 27, 10, 26, 25]. Fourth, the devised learning algorithm ORPegasos empowers RQDF to learn a better generalized query-to-combination-weights mapping with less run-time, which is a desirable property in large databases.

# 5. EXPERIMENTAL CONFIGURATION

In this section, we give a detailed description of the experimental configuration for empirical study. It includes three components: test collection, multimodal search experts, and methodology. Basic test procedures were designed following the guidelines of TRECVID [23] for performance evaluation of multimedia search systems. All experiments were conducted on a DELL PowerEdge 2970 workstation with 2 CPUs (each is a Quad-Core Intel Xeon E5420, 2x6MB cache CPU) and 32GB memory (DDR-2 667MHz).

## 5.1 Test Collection

A test collection was constructed with a large number of music items, the annotated ground truth, and a comprehensive set of social queries[5].

### 5.1.1 Music Items

To leverage the readily available online resources, the titles of the most popular 17000 music items related to the tag set in the social query space were automatically discovered from Last.fm based on its track popularity API. The actual audio track and metadata (title, description, tags, comments, etc.) of each music item were further crawled from YouTube using its open data API. The music data are diversely distributed due to the diversity of the social tags in the social query space. For each music dimension under consideration (i.e., genre, mood, instrument, and vocalness), ground truth data (the actual music style in each dimension) for the crawled music items were annotated and cross checked by amateur musicians with a reference to the popular social tags on Last.fm. The detailed data distribution over different music styles can be found in Table 3.

### 5.1.2 Social Queries

For evaluation, 236,973 social query topics for multimodal music search were automatically formed using the method described in Section 3. 200K of them were randomly selected as the training set, and the rest were used as the testing set of different QDF approaches. The query distribution over different music dimension combinations is shown in Table 4. As can be seen, the formed social queries range from less complex (only one music dimension is requested) to more complex (more music dimensions are requested) queries. This simulates the different levels of complexity in users' information need. In addition, we can see that the large number of automatically-formed social queries cover the knowledge scope of the social query space comprehensively, which lays the foundation for training a better generalized QDF system against unseen user queries.

## 5.2 Multimodal Search Experts

A text retrieval expert and a content-based audio retrieval expert were implemented for each of the four music dimensions being studied. Totally, 8 retrieval experts were used in the QDF evaluation. Each incoming query was parsed based on the social query space so that the music dimensions and

[5]The test collection can be obtained by emailing the first author.

**Table 3:** The detailed distribution of the music items in different music dimensions and styles. The number of collected music items is indicated after each style label. Some music items are shared by multiple music dimensions.

| **Genre:13747** | | **Mood:1596** | **Vocalness:2981** | **Instrument:1392** | | | |
|---|---|---|---|---|---|---|---|
| Classical:1359 | Jazz:1837 | Joyful:359 | Nonvocal:975 | Brass:310 | Woodwinds:382 | Percussion:356 | Strings:344 |
| Country:1954 | Rock:1417 | Pleasure:405 | Male:1025 | Trombone:103 | Flute:124 | Piano:129 | Violin:111 |
| Pop:2028 | HipHop:1062 | Sad:459 | Female:484 | Trumpet:104 | Clarinet:125 | Snare:100 | Cello:100 |
| Electronic:1987 | Metal:2103 | Angry:373 | Mixed:497 | Tuba:103 | Saxophone:133 | DrumKit:127 | Guitar:133 |

**Table 4:** The distribution of the automatically formed social queries over different music dimension combinations.

| **Query Type** | **No.** | **Example Queries** with "Find me the music with style" as the heading |
|---|---|---|
| Genre | 158 | classical; country; disco; jazz; metal; hip-hop; funk; black-metal; R&B; soul; romantic. |
| Mood | 173 | happy; excited; affective; hostile; flaming; aggressive; tearing; tense; fierce; playful. |
| Instrument | 309 | brass; trumpet; percussion; piano; drum-kit; strings; violin; woodwinds; flute; clarinet. |
| Vocalness | 13 | female; girl; woman; boy; male; man; mixed; nonvocal; instrumental. |
| Genre, Mood | 13165 | affective classical; hostile disco; flaming metal; excited country; joyful electronic. |
| Genre, Instrument | 13373 | classical violin; jazz saxophone; hip-hop drum; soul violin; metal clarinet; classical bagpipes. |
| Genre, Vocalness | 2546 | female country; mixed classical; nonvocal jazz; boy metal; girl pop; man electronic. |
| Mood, Instrument | 13783 | happy violin; hostile woodwinds; excited flute; affective piano; tearing trumpet. |
| Mood, Vocalness | 2951 | affective girl; happy boy; tearing mixed; hostile nonvocal; excited man; joyful girl. |
| Instrument, Vocalness | 4293 | violin female; drum male; violin girl; piano mixed; saxophone man; nonvocal guitar. |
| Genre, Mood, Instrument | 21395 | happy classical violin; tearing country guitar; fierce disco drum; joyful jazz saxophone. |
| Genre, Mood, Vocalness | 19209 | happy jazz female; affective classical male; aggressive metal mixed. |
| Genre, Instrument, Vocalness | 19305 | classical violin female; jazz saxophone male; hip-hop drum man; techno clarinet nonvocal. |
| Mood, Instrument, Vocalness | 19658 | excited violin boy; tearing piano girl; flame trumpet nonvocal; joyful guitar man. |
| Genre, Mood, Instrument, Vocalness | 106642 | happy classical violin female; tearing pop piano male; excited jazz saxophone nonvocal; flame metal drum mixed; affective country guitar man. |

actual music styles requested in the query can be recognized. The keyword of each music dimension in the query will be sent to the corresponding text retrieval and audio retrieval experts for unimodal search.

### 5.2.1 Retrieval based on Text

Text is a very important knowledge source for multimodal music search. For each music dimension, the text retrieval expert takes the corresponding keywords extracted from a social query and searches for relevant music items in their metadata. As described in Section 5.1.1, the metadata of a music item contains the title, description, tags, and comments. The textual metadata are stemmed using Porter's algorithm [17] and stop words are removed. The retrieval is done using the OKAPI BM-25 formula [18]. Our experiments show that the performance of the text retrieval expert varies in different music dimensions, because the keywords describing music styles may appear more often in some dimensions like genre than others like vocalness.

### 5.2.2 Retrieval based on Audio Content

Audio content could play a crucial role in music search. In the audio retrieval expert of each music dimension, a compact audio signature called Fuzzy Music Semantic Vector (FMSV) is constructed as the content representation using Multi-class Support Vector Machines to classify the audio features presented in [28]. The audio signature is formed by the activation probabilities of the music classes in this dimension. For example, 8 activation probability values of the genre classes form a genre audio signature. To improve scalability and efficiency, a high dimensional index of audio signatures is constructed using the Locality Sensitive Hashing (LSH) algorithm [1]. When a keyword describing a music style is extracted from a social query, a query audio signature $\mathbf{q}$ can be generated by setting the activation probability of the corresponding music class as 1 and other classes as 0. Nearest neighbors of $\mathbf{q}$ are then retrieved by LSH from the index. Further ranked by the inverse value of the Euclidean distance from an audio signature to $\mathbf{q}$, the most similar music items are retrieved to the music style described by the keyword of a music dimension in the social query. In this study, we implemented the audio analysis based on Marsyas [24], which is the system having achieved the best performance in the corresponding classification tracks of MIREX2008 [6].

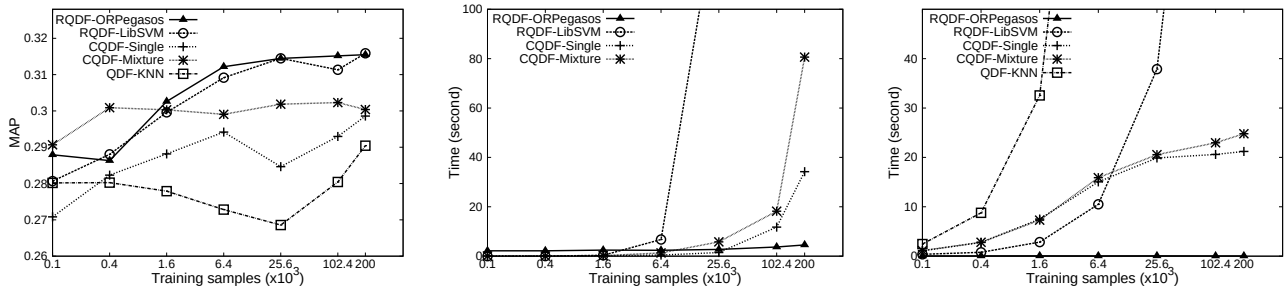
### 5.2.3 Score Normalization, Fusion, and Relevance Judgment

To achieve a more robust ranking process, the scores $\{s_{i,d_j} | 1 \leq j \leq N_r\}$ for the $i$-th ranked list of music items are normalized as $s_{i,d_j} = 1 - Rank_i(d_j)/N_r$, where $N_r$ is the maximum number of music items returned by each retrieval expert (set as 100 in our experiment) and $d_j$ indicates the unique id of the $j$-th music item [27]. When a weight vector $\mathbf{W} = [W_1, ..., W_N]^T$ is available, the final ranking scores of the music items from all retrieval experts can be linearly combined as $s_d = \sum_{i=1}^{N} W_i \cdot s_{i,d}$, where $N$ is the number of search experts, $0 \leq W_i \leq 1$, and $\sum W_i = 1$.

The relevance of each music item in the final ranked list is judged by comparing its annotated music styles (described in Section 5.1.1) with the music styles requested in a social query. If a music item matched $m$ out of $n$ music styles in a social query, the relevance score can be set as $r = m/n$. Since users tend to issue queries with orthogonal information need in different music dimensions (e.g., Find me music with style classical female), the fraction relevance score introduced in this work is necessary to reflect the partial match of a music item to a social query and guarantee a more accurate measurement of different search methods.

## 5.3 Methodology

The goal of our empirical study is to investigate the performance of the proposed algorithm and its competitors from different aspects. To evaluate effectiveness of various systems, Average Precision (AP) is applied to measure the retrieval accuracy of a ranked list at depth 100. We also use Mean Average Precision (MAP) to evaluate the retrieval accuracy of a QDF method over a set of social queries,




(a) Retrieval accuracy (MAP) (b) Training time (c) Prediction time

**Figure 3:** The comparison of different QDF methods in terms of effectiveness and efficiency.

and $t$ test to assess the significance of performance improvement between different QDF methods. We have fully implemented the previous QDF methods for comparison study. They include: the CQDF method based on a single-class query matching (CQDF-Single) with learning on the oracle APs or combination weights [10]; a mixture-of-classes query matching (CQDF-Mixture) based on the oracle combination weights [26]; together with the nearest-neighbor query matching approach (QDF-KNN) based on the oracle combination weights [25]. To further illustrate the superior efficiency of ORPegasos in RQDF, we also implemented a LibSVM [2] version of RQDF (RQDF-LibSVM).

For each social query, the AP of each search expert was calculated. Grid search was used to compute the oracle combination weights. A social query plus its oracle combination weights form a training sample for different QDF methods. Different training set sizes were used (100, 400, 1.6K, 25.6K, 102.4K and 200K) to evaluate the impact of the training knowledge scope on the retrieval performance. For a particular training set size, training samples were randomly selected from the total 200K training queries to train all the QDF methods. The trained QDF model was validated against all testing social queries (about 36.9K). Three trials were conducted for each QDF method to assess its average performance. The average APs or MAPs were collected to compare the retrieval accuracy. The average run-time of each QDF method in training a query-to-combination-weights mapping and in predicting the combination weights of a common query set were used to measure the efficiency.

# 6. RESULT ANALYSIS

## 6.1 Effectiveness Study

Retrieval accuracy is one of the main concerns in any IR system. In this section, we study and compare the effectiveness of different QDF methods when different number of training samples are available.

Fig. 3(a) and Table 5 illustrate the retrieval accuracy (MAP) of different QDF methods under their best parameter settings. 40 classes ($C$=40) were discovered on oracle APs in CQDF-Single. Learned on combination weights, 50 classes ($C$=50) and a mixture of top 10 classes ($T = 10$) were chosen in CQDF-Mixture to match an unseen query. QDF-KNN used the top 5 nearest neighbors ($K = 5$) to represent a user query. Linear kernels were used for RQDF-LibSVM and RQDF-ORPegasos. 5 training samples were chosen ($k = 5$) at each iteration of ORPegasos. From these results, the following two trends can be discovered.

*More training queries produce better generalization performance.* In Fig. 3(a), as more training queries become available to each algorithm, the retrieval accuracy gets improved. The previous QDF methods (CQDF-Single, CQDF-Mixture, and QDF-KNN) are improved by absolute (relative) 2.8% (10.4%), 0.9% (3.1%), and 2.2% (8.2%) in MAP respectively, when the number of training queries increases from 100 to 200K. The fluctuation of their MAPs is due to the poor robustness, which will be discussed in Section 6.3. The two versions of the regression-QDF (RQDF-LibSVM and RQDF-ORPegasos) are improved by 3.5% (12.46%) and 2.8% (9.8%) respectively under the same condition. In Table 5, the retrieval accuracy of each method on different query types shows the same trend, especially in query types with more information need (more complex queries). Since the training queries are randomly sampled from the social query space, more training queries cover a wider knowledge scope of unseen user queries. QDF methods trained with more query knowledge reveal better generalization performance. Therefore, these results support our hypothesis that as the knowledge scope of the training queries gets closer to the unseen user queries, the generalization performance of QDF approaches will be improved. This demonstrates the significance of automatically forming a large number of comprehensive social queries to better develop QDF systems.

*Regression-QDF (RQDF) outperforms other methods.* In Fig. 3(a), both versions of RQDF outperform previous QDF methods except CQDF-Mixture with a small number of training samples. As more training samples become available, RQDF reveals more improvement in the retrieval accuracy over other methods. When using 200K training samples, RQDF outperforms other methods by about absolute (relative) 1.3% (4.3%) in MAP. Between the two versions of RQDF, ORPegasos matches the performance of LibSVM. The MAPs in Table 5 also illustrate the superior retrieval accuracy of RQDF. In most query types, RQDF performs the best. RQDF reveals more improvement as the query becomes more complex. The significance of the effectiveness improvement by RQDF over other methods is verified by $t$ test. The $p$-value is less than 0.05, which indicates that the effectiveness improvement of RQDF is statistically significant. These results demonstrate the superior modeling capability of RQDF to learn and predict the optimal combination strategy.

## 6.2 Efficiency Study

With increasing attention in academia, efficiency is another concern in real-life IR applications. In this section, we compare the training time and prediction time of different QDF methods. Less training time enables faster system de-

**Table 5:** The retrieval accuracy (MAP) of each QDF method in different query types. CQDF-Mixture-Weight used 10 mixture classes ($T$=10). **No. (∗)** indicates the number of training queries, $* = \times 10^3$. G, M, I, V indicate the four music dimensions (genre, mood, instrument and vocalness). **A** indicates all the four dimensions. Bold font indicates the best MAP across all training sets of the same method. $^\dagger$ indicates the best MAP across all methods.

| Method | No. (∗) | A | G | M | I | V | GM | GV | GI | MI | MV | IV | GMI | GMV | GIV | MIV | GMIV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CQDF-Single-AP, $C$=40 | 0.1 | .270 | **.307** | .439 | .850 | .525 | .271 | .289 | .341 | .334 | .367 | .399 | .235 | .249 | .283 | .306 | .229 |
| | 1.6 | .287 | .305 | .446 | **.851** | .527 | .291 | .303 | .376 | **.396** | .367 | .434 | .257 | .255 | .304 | .320 | .238 |
| | 25.6 | .284 | .303 | .445 | .810 | **.536** | **.311** | .306 | .358 | .358 | .372 | .422 | .255 | .259 | .300 | .313 | .237 |
| | 200 | **.298** | .304 | **.450** | .830 | .525 | .308 | **.310** | **.391** | .385 | **.373** | **.451** | **.265** | **.260** | **.319** | **.331** | **.250** |
| CQDF-Single-Weight, $C$=40 | 0.1 | .278 | .303 | .442 | .850 | **.541** | .278 | .285 | .385 | .377 | .360 | .398 | .258 | .246 | .283 | .304 | .232 |
| | 1.6 | .290 | .298 | .442 | .850 | .525 | .298 | .285 | .419 | .406 | .371 | .421 | .274 | .247 | .290 | .323 | .237 |
| | 25.6 | .291 | **.312** | .444 | **.855**$^\dagger$ | .532 | **.314**$^\dagger$ | .295 | .394 | .399 | **.374** | .414 | .276 | .257 | .289 | .325 | .239 |
| | 200 | **.302** | .300 | **.445** | .854 | .522 | .309 | **.295** | **.438** | **.434** | .371 | **.421** | **.293** | **.257** | **.302** | **.327** | **.249** |
| CQDF-Mixture-Weight, $C$=50 | 0.1 | .290 | .304 | .441 | .850 | .537 | .278 | .272 | .410 | .413 | .374 | .445 | .272 | .239 | .305 | .325 | .238 |
| | 1.6 | .299 | .308 | .441 | .850 | **.554** | .287 | .309 | .421 | .408 | .378 | **.446** | .277 | **.266** | **.315** | **.329** | **.247** |
| | 25.6 | **.301** | .310 | **.449** | .851 | .544 | **.309** | .310 | **.430** | **.425** | .380 | .427 | **.289** | .264 | .305 | .326 | .246 |
| | 200 | .299 | **.310** | .444 | **.851** | .541 | .301 | **.310** | .426 | .418 | **.380** | .426 | .283 | .265 | .304 | .327 | .246 |
| QDF-KNN, $K$=5 | 0.1 | .268 | .307 | .440 | .850 | .550 | .279 | .296 | .361 | .383 | .345 | .395 | .248 | .238 | .272 | .290 | .218 |
| | 1.6 | .277 | .309 | .441 | .852 | **.560**$^\dagger$ | .292 | .308 | .394 | .376 | .352 | .399 | **.260** | .248 | .284 | .300 | .225 |
| | 25.6 | .270 | .306 | **.454**$^\dagger$ | .853 | .525 | .300 | .310 | **.394** | **.388** | .346 | .370 | .250 | .245 | .263 | .281 | .219 |
| | 200 | **.290** | **.313**$^\dagger$ | .449 | **.854** | .525 | **.309** | **.332**$^\dagger$ | .389 | .364 | **.381** | **.440** | .250 | **.270** | **.302** | **.320** | **.239** |
| RQDF-ORPegasos, $k$=5 | 0.1 | .287 | **.307** | **.442** | .850 | **.559** | .280 | .298 | .423 | .419 | .362 | .425 | .276 | .246 | .294 | .312 | .231 |
| | 1.6 | .302 | .306 | .439 | **.851** | .525 | .302 | .317 | .432 | .425 | .377 | .451 | .283 | .269 | .314 | .330 | .244 |
| | 25.6 | .314 | .303 | .441 | .850 | .527 | **.306** | .321 | .440 | .438 | .385 | .458 | .299 | .279 | .325 | .342 | .258 |
| | 200 | **.315**$^\dagger$ | .301 | .441 | .850 | .524 | .305 | **.323** | **.440**$^\dagger$ | **.438**$^\dagger$ | **.388**$^\dagger$ | **.460**$^\dagger$ | **.299**$^\dagger$ | **.281**$^\dagger$ | **.326**$^\dagger$ | **.344**$^\dagger$ | **.259**$^\dagger$ |
| Oracle | N.A. | .337 | .393 | .521 | .874 | .787 | .354 | .368 | .451 | .451 | .417 | .485 | .313 | .311 | .344 | .365 | .280 |

velopment and upgrade while less prediction time increases the system scalability in accommodating a large number of online queries simultaneously.

The run-time (training time and prediction time) of different QDF methods are illustrated in Fig. 3(b) and Fig. 3(c). As can be seen, the run-time of RQDF-ORPegasos is much less than other methods in both training and prediction stages when different number of training queries are used. In addition, as the number of training queries increases, the run-time of RQDF-ORPegasos stays almost constant while the run-time of other QDF methods increases dramatically.

The results can be explained by the learning model employed by each QDF method. When more training queries are used, class-based QDF (CQDF-Single/CQDF-Mixture) requires more training time to discover classes by weight clustering [10] or latent query class analysis [26]. Their prediction time also increases as there are more non-zero elements in the discovered query-class centroids, which are used to assign an unseen query to a query-class (query-classes) and derive the predicted combination weights. QDF-KNN requires no training time since the combination weights are predicted by linearly embedding an unseen query into its nearest neighbors in the raw training set. However, its prediction time increases dramatically when a large training set is used due to the time consuming linear search of nearest queries to an unseen query. Compared with the previous QDF approaches, RQDF-ORPegasos/RQDF-LibSVM eliminates the query matching step. However, the run-time of RQDF-LibSVM suffers in large training sets because the SVM model tends to discover and use more support vectors to represent the query-to-combination-weights mapping. As analyzed in Section 4, the run-time of ORPegasos is independent of the number of training samples. Analyzed and verified in [22], ORPegasos will reveal reduced run-time in extremely large training data sets when learning SVR models with the same generalization performance.

The scalability of QDF-ORPegasos on large data sets is superior to other QDF methods. Based on the run-time of different QDF methods, RQDF-ORPegasos is suitable for efficient training of an effective QDF system in real-life applications where a large volume of training data are available. In addition, due to its efficient prediction time, RQDF-ORPegasos will scale well when a large number of online users queries needs to be processed simultaneously.

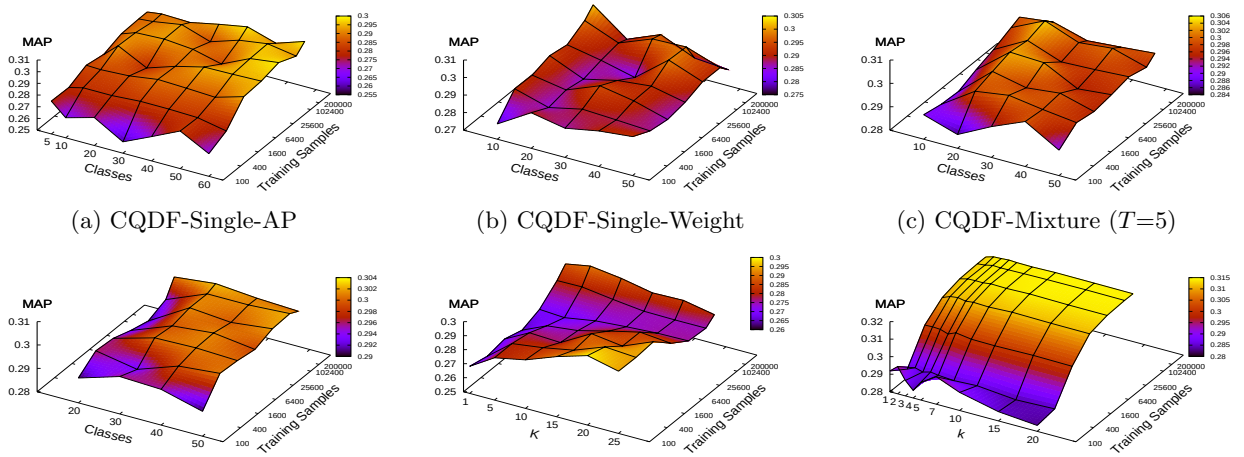
## 6.3 Robustness Study

As mentioned in Section 6.1, poor robustness is one disadvantage of the previous QDF methods, which makes the system tuning tricky. In this section, we compare the robustness of different QDF methods by investigating the effect of different parameter settings on their retrieval accuracy.

Fig. 4 illustrates the retrieval accuracy of different QDF methods with various parameter settings. As can be seen, the retrieval accuracy of RQDF (RQDF-ORPegasos) is much more stable than other methods. Under all parameter settings, ORPegasos outperforms other methods when sufficient training samples are available. In addition, its peak performance always appears with the maximum number of training samples. Due to its stochastic nature, the parameter $k$ of ORPegasos has little influence on its generalization performance. Although fewer number (when $k$ is small) of training samples are chosen at each iteration to update the SVR model, the expectation of the model converges to the global optimal over $T$ iterations (see Section 4 for details).

However, the effectiveness of other methods fluctuates severely under different parameter settings. In particular, QDF-KNN reveals two peak performance regions in Fig. 4(e). The region with less training samples and larger $K$ performs well because using more nearest neighbors to match a user query results in better robustness when less training samples are available. However, as more training samples become available to cover a wider knowledge scope of unseen user queries, using less nearest neighbors (smaller $K$) will increase the query matching accuracy. This also explains the MAP fluctuation of QDF-KNN in Fig. 3(a).

All these results show that it is easier to tune a QDF system with RQDF than with other methods, which means RQDF has higher usability and better robustness in real-life applications. In addition, the superior retrieval accuracy




(a) CQDF-Single-AP (b) CQDF-Single-Weight (c) CQDF-Mixture ($T$=5)

(d) CQDF-Mixture ($T$=10) (e) QDF-KNN (f) RQDF-ORPegasos

**Figure 4:** The retrieval accuracy comparison of different QDF methods under various parameter settings.

of RQDF and the overall trend that more training samples produce better effectiveness are verified once more in Fig. 4.

# 7. CONCLUSION

We have outlined a novel query-dependent fusion (QDF) method using regression-on-folksonomies to facilitate multimodal music search in large databases. Previous QDF approaches rely on manually designed queries, which imposes expensive human involvement in system development. We have pursued the alternative automatic query formation to easily generate a large number (in millions) of comprehensive queries from readily available online folksonomy data. This approach not only provides better generalization performance for real-life search systems in accommodating future user queries, but also offers great feasibility and practicality in real-life system development.

In addition, we have proposed an online-regression model for query-dependent fusion (RQDF). It represents a further step towards the optimal query-dependent fusion, which unleashes the power of multimodal music search. Its superior modeling capability not only enhances the effectiveness of query-dependent fusion systems, but also significantly improves the system efficiency, scalability, and robustness. Due to the generality of RQDF, we believe that it can be easily extended to other multimodal search applications such as text/video/image search and meta-search as well.

# 8. ACKNOWLEDGEMENT

This work is supported by the research grant Multimodal Mobile Music Retrieval with WBS no. R-252-000-381-112.

# 9. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS'06*, 2006.

[2] C. Chang and C. Lin. Libsvm: a library for support vector machines, 2001.

[3] T.-S. Chua, S.-Y. Neo, K.-Y. Li, G. Wang, R. Shi, M. Zhao, and H. Xu. Trecvid 2004 search and feature extraction task by nus pris. In *NIST TRECVID Workshop*, 2004.

[4] B. Cui, L. Liu, C. Pu, J. Shen, and K. L. Tan. Quest: querying music databases by acoustic and textual features. In *ACM Multimedia'07*, 2007.

[5] C. V. Damme, M. Hepp, and K. Siorpaes. Folksontology: An integrated approach for turning folksonomies into ontologies. In *the ESWC Workshop*, 2007.

[6] S. J. Downie. The music information retrieval evaluation exchange (mirex). In *ISMIR'08*, 2008.

[7] N. R. Draper and H. Smith. *Applied Regression Analysis*. Wiley-Interscience, 1998.

[8] I.-H. Kang and G. Kim. Query type classification for web document retrieval. In *SIGIR '03*, 2003.

[9] L. Kennedy, S. F. Chang, and A. Natsev. Query-adaptive fusion for multimodal search. *Proc. of the IEEE*, 2008.

[10] L. Kennedy, A. P. Natsev, and S. F. Chang. Automatic discovery of query-class-dependent models for multimodal search. In *ACM Multimedia'05*, 2005.

[11] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Trans. on Signal Processing*, 2004.

[12] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[13] Q. Mei, J. Jiang, H. Su, and C. Zhai. Search and tagging: Two sides of the same coin? Technical report, 2007.

[14] P. Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semantics*, 2007.

[15] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 1995.

[16] X. Olivares, M. Ciaramita, and R. van Zwol. Boosting image retrieval through aggregating search results based on visual annotations. In *ACM Multimedia'08*, 2008.

[17] M. F. Porter. An algorithm for suffix stripping. *Program*, 1980.

[18] S. E. Robertson, S. Walker, M. M. Beaulieu, and M. Gatford. Okapi at trec-4. In *TREC-4*, 1995.

[19] S. S. Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML'07*, 2007.

[20] B. Schölkopf and A. J. Smola. *Learning with Kernels*. Cambridge, MA: MIT Press, 2001.

[21] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *TREC-2*, 1994.

[22] S. Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *ICML'08*, 2008.

[23] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06*, 2006.

[24] G. Tzanetakis and P. Cook. marsyas a framework for audio analysis. *Organized Sound*, 2000.

[25] L. Xie, A. Natsev, and J. Tesic. Dynamic multimodal fusion in video search. In *IEEE ICME'07*, 2007.

[26] R. Yan and A. G. Hauptmann. Probabilistic latent query analysis for combining multiple retrieval sources. In *SIGIR'06*, 2006.

[27] R. Yan, J. Yang, and A. G. Hauptmann. Learning query-class dependent weights in automatic video retrieval. In *ACM Multimedia'04*, 2004.

[28] B. Zhang, J. Shen, Q. Xiang, and Y. Wang. Compositemap: A novel framework for music similarity measure. In *Proc. of ACM SIGIR*, 2009.