# User-specific Training of a Music Search Engine

**David Little, David Raffensperger, Bryan Pardo**
**Electrical Engineering and Computer Science**
**Northwestern University**
**Evanston, IL, USA**
**{d-little, d-raffensperger, pardo}@northwestern.edu**

## Abstract

Query-by-Humming (QBH) systems let a user find the desired song in a music database by humming or whistling its melody. Existing systems do not optimize on individual users, once deployed. We present a method to improve QBH performance with user-specific training on a deployed system. Parameters for the singer error model and note transcription are tuned using a genetic algorithm. Testing over a corpus of sung queries [3] our preliminary tests show songs within and near the top ten songs listed.

## 1. Introduction

People often wish to find a piece of music but do not know the name or artist of a piece. Examples include artists who want to address or avoid copyright infringement, or casual listeners who can recall the melody but not the name of the desired song. Query-by-Humming (QBH) systems [6] let a find a song from a database, based on a sung or hummed query.

Each user will bring different aptitudes and weaknesses to the task of singing a melody. Thus, systems must be able to handle a variety of different singing styles. This can be facilitated through system training. Several QBH systems have utilized some form of generalized training for the tasks of transcription and comparison. For example, Meek and Birmingham [4] train a detailed a model of singer error for a Hidden Markov Model. Parker, Fern and Tadapalli [5] propose a general model for learning string matching alignments through the boosting of learners such as support vector machines or neural networks. Such systems typically require the structure of the queries be annotated before training can begin. For example, note segmentation requires segmented examples. Also, these systems are not designed for user-specific training after system deployment. We have built a QBH system that personalizes a singer model based on user feedback, learning the model on-line, after deployment. The more a person uses and corrects the system, the better the system performs.

The system is outlined in Figure 1. The user sings a query(1). The system returns a list of songs from the database, ranked by similarity(2). The user listens to the songs returned and selects the desired one(3). The recorded query-to-correct-target paring is saved to a database(4). This database is used to train parameters for melody transcription and query comparison(5).

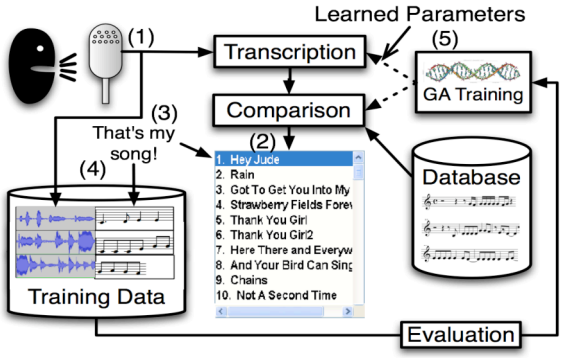


**Figure 1**: Our model of QBH training.

The system tunes parameters by optimizing on the average rank of the correct targets for a set of stored queries, such as all recent queries by a single user. This can take place automatically with a deployed system, allowing personalization after deployment with no need for operator intervention to hand-label melodic transcriptions as "ground truth" comparisons.

## 2. Learned Parameters

The system currently learns two sets of parameters: those for query transcription and those for the singer error model. Each melody in our database of targets is encoded as a sequence of pitch and rhythm intervals between adjacent notes [1]. We encode our sung queries similarly. The singing is first transcribed into a sequence of fixed-length 10 millisecond frames. Each frame is represented by three values: pitch, harmonicity and amplitude. The sequence of frames is segmented into notes through local thresholding on the difference between adjacent frames. This results in four learnable parameters: the relative weight given to each dimension (pitch, harmonicity, amplitude) and the chosen threshold.

Once a query has been turned into a sequence of note intervals, the system must determine its similarity to each target song in the database. Popular methods include string-matching, n-grams and Markov models [1]. Results in [1] show probabilistic string-matching is a successful approach; it is the method we use. String-matching requires a reward function for comparing sequence elements. The more similar two elements are, the higher the reward for calling them a match. We encode a singer error model in this function. For example, the more often a singer confuses

major thirds with minor thirds, the less we should penalize a difference between a major third in the query and a minor third in the target.

Our error model uses a parameterized function defining the similarity between note intervals. The function takes five parameters, the weight of pitch and rhythm, the precision of each of these and the degree of octave similarity.

# 3. Learning Method

We learn the four parameters for note segmentation and five parameters for note interval similarity with a genetic algorithm [7]. Parameters are represented as a binary fraction with a total of 7 bits ranging from zero to one (by units of 1/127). Because the weights within each function (note segmentation and interval similarity) are dependent on each other, we can fix a weight per function, giving us a total of seven different parameters to be learned. We allow a six point crossover (between each of the seven parameters). All parameters for each function were adjacent.

During each generation, the fitness of each individual is found using the Mean Reciprocal Rank (MRR) of the correct target. (So if the correct target is generally third in the list of songs, its MRR would be ~1/3). This fitness was used to determine which half of the population would produce children for the next generation using fitness proportional reproduction.

# 4. Empirical Tests

For preliminary tests we chose a population size of 30, running for a total of 15 generations. The initial test took approximately 3 days. Our more extensive test, after optimizations, is expected to run for approximately a week.

The target database consists of 1049 melodies drawn from classical music and folk songs. A chance MRR, assuming a uniform distribution of rankings is log(n)/n = 0.0066[1]. Our query data was a set of 6 songs each sung by 4 singers, for a total of 24 training queries. These queries were drawn from a QBSH corpus [3] used during the 2006 MIREX comparison of query-by-humming systems [2]. Ou

We wanted to ask two questions during testing: how well can we generalize to other songs after user training, and how well can we generalize to other users after training on a population (general training)? During user training, we trained on five of the queries sung by a user and tested on the sixth, repeating this for each of the six queries. During general training, we trained on all songs for three of the singers and tested with the fourth singer's songs, repeating this for each singer:

Table 1 summarizes our results for the preliminary test. Mean MRR's were largely within or near the top ten (1/10 = 0.1) songs. Given the added statistical power of our full experiment we will be able to make a more thorough analysis of the final results.

**Table 1:** The preliminary test's results.

| | Mean Reciprocal Rank | | | |
|---|---|---|---|---|
| | **User Trn** | **User Test** | **Gen Trn** | **Gen Test** |
| **Mean** | 0.273 | 0.0488 | 0.158 | 0.0897 |
| **SD** | 0.17 | 0.40 | 0.012 | 0.097 |

# 5. Conclusions

Our model for training would allow a QBH system to train without the use of query annotations: it thus has the potential to be used to improve system performance after deployment, allowing us to tune parameters for specific users. With our fairly limited initial test we were able to show mean MRR's near the top ten songs, and we will be running more substantial tests shortly.

# References


[1] R. Dannenberg, W. Birmingham, B. Pardo, N. Hu, C. Meek and G. Tzanetakis, *A Comparative Evaluation of Search Techniques for Query-by-Humming Using the MUSTART Testbed*, Jounral of the American Society for Information Sicence and Technology, 58 (2007).

[2] J. S. Downie, K. West, A. Ehmann and E. Vincent, *The 2005 Music Information retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview*, *6th International Conference on Music Information Retrieval*, September 11-15, London, UK, 2005.

[3] Jyh-Shing and R. Jang, *QBSH: A corups for Designing QBSH (Query by Singing/Humming) Systems*, 2006.

[4] C. Meek and W. Birmingham, *The dangers of parsimony in query-by-humming applications*, *International Conference on Music Information Retrieval*, Washington, DC, 2003.

[5] C. Parker, A. Fern and P. Tadepalli, *Gradient boosting for sequence alignment*, *The Twenty-First National Confeerence on Artificial Intelligence*, Boston, MA, 2006.

[6] R. Typke, F. Wiering and R. C. Veltkamp, *A Survey of Music Information Retrieval Systems*, *6th International Conference on Music Information Retrieval*, London, UK, 2005.

[7] A. Wright, *Genetic algorithms for real parameter optimization*, *The First workshop on the Foundations of Genetic Algorithms and Classier Systems*, Bloomington, Indianna, 1990.