BUILDING A MUSIC SEARCH DATABASE USING HUMAN COMPUTATION

Mark Cartwright and Bryan Pardo
Northwestern University
Department of Electrical Engineering and Computer Science
2133 Sheridan Road, Evanston, IL 60208, USA.
mcartwright@u.northwestern.edu, pardo@cs.northwestern.edu

ABSTRACT

Systems able to find a song based on a sung, hummed, or whistled melody are called Query-By-Humming (QBH) systems. Hummed or sung queries are not directly compared to original recordings. Instead, systems employ search keys that are more similar to *a cappella* singing than the original pieces. Successful, deployed systems use human computation to create search keys: hand-entered midi melodies or recordings of *a cappella* singing. There are a number of human computation-based approaches that may be used to build a database of QBH search keys, but it is not clear what the best choice is based on cost, computation time, and search performance. In this paper we compare search keys built through human computation using two populations: paid local singers and Amazon Mechanical Turk workers. We evaluate them on quality, cost, computation time, and search performance.

1. INTRODUCTION

Music audio is a popular category of multimedia content. Services like iTunes provide millions of songs to the public, but typically index their recordings with such metadata as title, composer, and performer. Finding the desired recording can be a problem for those who do not know the metadata for the desired piece.

One solution is to identify a song based on entering its lyrics into a standard text-based search engine. This, however, does not apply to instrumental music. If the user has access to a recording of the desired audio (e.g. it is currently playing on the radio), then an audio fingerprinting system [1, 2] can be used. If the user cannot provide a portion of the exact recording sought (e.g. the song ended on the radio), such systems cannot help. If the user can sing or hum some portion of the song, a query-by-humming (QBH) [3] system can be used.

Most published research in QBH [4–6] has focused on the matching algorithms and distance measures for melodies. These are not, however, the only challenges that must be surmounted to build an effective QBH system ready for real-world deployment. One key challenge for current systems is the creation of a large database (perhaps millions) of relevant search keys that are effective for matching against sung or hummed queries.

Creating searchable keys that can be queried by singing is non-trivial. Sung queries typically outline a melody drawn from the desired recordings. The vast majority of music recordings do not have machine-readable notated scores or MIDI versions available. Therefore, melodic keys must be created directly from the audio. Currently, automated approaches to extracting the main melody from a polyphonic recording are not sufficiently robust to build melodic keys from large databases of music recordings [7].

Deployed QBH systems, such as Soundhound [8] or Tunebot [9] have attempted to solve this problem through the use of human computation. Human computation is a technique where a computational system outsources certain steps to humans. The database for Tunebot, for example, uses searchable melodic keys derived from *a cappella* performances contributed by users through use of the Tunebot search engine (i.e. matched queries are converted to search keys), and by logging in as a contributor and singing melodies to the system. However, systems are said to suffer from the *cold start problem* if their functionality depends on usage data from the same users they are serving (the classical example of this is recommendation systems). QBH systems that depend on user contributions as search keys also suffer from this. Therefore, before obtaining users, QBH systems must first overcome the cold start problem by having a sufficiently large database to attract users to use the system. One way to overcome that problem is to hire vetted, local, paid singers to create searchable melodic keys. While this method may likely yield well-sung, highly-controlled search keys, it is costly. Amazon's Mechanical Turk, a micro-task labor market, may provide a cheaper solution, but how do the resulting search keys compare to those generated by vetted, local singers? Would the quality of the search results be affected? Would we need more of them to compensate for the potentially reduced quality control? Which method can build a database faster? Which method can build a database cheaper?

In this paper we compare search keys built through human computation using two populations: paid local singers and Amazon Mechanical Turk (M.Turk) workers. We evaluate the difficulty of creating a searchable database using both methods and the effectiveness of the resulting

database. In a previous workshop paper [7], we compared the two human computation methods to two promising machine computation methods [10,11], but found that the machine computation methods still perform much worse than human computation. In this paper we extend that work by focusing strictly on comparing the human computation methods, providing more thorough descriptions of the two methods as well as comparisons on cost, completion time, and performance. Additionally, we analyze how the number of search keys affect system performance.

The remainder of the paper is structured as follows: Section 3 describes the search engine used to perform comparisons. Section 4 describes two approaches taken to building search keys with human computation. Sections 5-6 describes our two experiments. Section 7 contains conclusions.

2. BACKGROUND

There are two parties in human computation: the *requesters* (the party "requesting" that a human computation task be performed) and the *workers* (the party that performs the human computation task). According to Law and von Ahn [12], there are 6 recognized markets (pools of workers) for human computation: *gamers* (e.g. games with a purpose (GWAP), ESP Game [13]), *citizen science* (e.g. Galaxy Zoo [14]), *security and access* (e.g. reCAPTCHA), *temporary markets* (brief peaking markets caused by urgent circumstances such as natural disaster), *learners* (e.g. Duolingo [15]), and *paid crowdsourcing* (e.g. Amazon Mechanical Turk). These markets have varying degrees of intrinsic/extrinsic motivations, risks, costs, etc. The best market to use is dependent on the problem that needs to be solved.

Some, but not all, of these markets have previously been explored for the generation of QBH search keys. For instance, the *gamers* market was explored in [9,16,17]. However, games have their drawbacks. While the players (workers) are not paid, the costs of designing, developing, and testing a game are high. It is also risky. If the game isn't fun and doesn't garner a player base, then few human computation tasks will be solved. Researchers have also explored other markets. The "citizen scientist" approach is similar to the user contribution / "citizen singer" approach that both Tunebot and SoundHound employ. This approach again works well if you can attract users to your site, but this may be difficult without a useful working system and therefore doesn't solve the cold start problem. The *security and access* approach has not been explored, but it seems like a poor approach for QBH search key generation task due to the time investment required for a task of its granularity (on the order of minutes). Additionally, unless earworms [1] become an international epidemic, it seems unlikely that a temporary market would arise from an urgent need to create QBH search keys. The *learners* market is a newly proposed human computation market that seems promising for future work, but one that would likely have high design and development costs similar to a GWAP.

[1] An *earworm* is a piece of music that seems to be "stuck on repeat" in one's mind.

In this paper, we chose to focus on the last group, the *paid crowdsourcing* market. We compare the human computation by M. Turk workers and that of locally hired (non-crowdsourced), vetted, paid singers.

3. THE TUNEBOT SYSTEM

To compare the search key generation methods in a realistic, real-world way, we insert search keys built using the methods described in this paper into the database of the QBH system, Tunebot [9]. Tunebot has been deployed on the web in its current form for over three years and has logged over 60,000 user queries. We now give a brief overview of Tunebot. For more detail, please see [9].

3.1 Query and Search Key Encoding

Before a melodic comparison takes place between a sung query and the search keys in the database, the transcriber estimates the fundamental frequency of the singing every 20 milliseconds using a pitch tracker based on PRAAT [18]. A note segmenter then divides this series of estimates into notes. We encode all queries and all melodies in the database (search keys) as sequences of note intervals.

Each note interval is represented by a pair of values: the pitch interval (PI) between adjacent notes (measured in units of musical half-steps) and the log of the ratio between the length of a note and the length of the following note (LIR). Note lengths are defined to be inter-onset-intervals. We use note intervals encoded in this way because they are transposition invariant (melodies that differ only in key appear the same) and tempo invariant (melodies that differ only in tempo appear the same).

3.2 Melody Matching

Once the query is encoded as a note interval sequence, it is compared to the search keys in the database, which have been similarly encoded. To compare melodic strings, we use edit distance [19]. The edit distance between two strings is the cost of the least expensive way of transforming one string into the other. The edit cost for a substitution is determined by the likelihood of that substitution, based on prior user query data. The pieces in the database are ranked by the edit distance of their search keys to the sung query, and this ranking is returned to the user.

4. HUMAN COMPUTATION OF QBH SEARCH KEYS

As stated earlier, in this paper we compare the generation of QBH search keys using human computation of M. Turk workers and that of locally hired (non-crowdsourced), vetted, paid singers. We had previously collected thousands of sung search keys from the paid local singers, and Mechanical Turk seemed the most viable human computation market to compare to. To effectively compare the two methods, we needed to generate two databases that had a minimum number of search keys per song across a fixed set of songs. We already had such a database from the locally hired singers, and the Mechanical Turk market gave

us the control we needed to generate the another example database using crowdsourced human computation.

4.1 The Song Set

The current Tunebot music database consists of search keys generated both from user contributions and from paid local singers. We chose the 100 songs with the most contributions in the current Tunebot music database as the set of target songs for comparison. These songs were mostly of popular music genres distributed as follows: 41% pop, 34% rock, 6% dance, 4% hip hop, 3% country, 12% other. With each human computation method and for each song, we collected at least 6 sung examples to be converted into search keys as described in Section 3.1.

We now describe the two human computation methods we used for the generation of QBH search keys for these target songs.

4.2 Hiring Paid Local Singers

As noted earlier, we had previously collected thousands of sung queries over the course of two years from paid local singers. With this approach, we had a lot of control over the quality of the search keys, but having that control came with a higher cost per search key.

We solicited singers through flyers posted throughout a college campus and on student online job listings. Candidates were interviewed and auditioned for singing ability by a trained musician with a graduate degree in music. The singers were hired based on their singing ability and employment availability.

To generate the sung examples, the singers recorded themselves in a sound isolation booth into a large diaphragm condenser microphone connected to a computer interface. They used the Tunebot web application to record themselves, which uploaded the recorded content to our server. They were primarily self-directed, following a set procedure, which ensured that they both sang the songs we wanted and that the quality of the sung examples was high and consistent. For a given song, an employed singer was instructed to sing the most memorable portions of the song, typically the verse, chorus or both. In addition, they were instructed to sing each memorable portion twice - once with lyrics and once without lyrics, since the segmentation of the note interval representation may differ when sung with lyrics versus without lyrics. Each song was assigned to one male singer and one female singer.

4.2.1 Computation Time and Cost

It took each singer a mean time of 4.26 (SD 3.79) minutes to learn and sing each example. Since we paid the singers $9.00/hr, each example cost about $0.64. It took roughly 9.5 days to create at least 6 sung examples for the 100 songs in the song set. Thus, they generated about 442 examples per week. When seeking 6 examples per song (3 male, 3 female), the computation time is approximately 26 minutes per song, and the cost of each song was approximately $3.90.

While the singers were generally reliable, they did require some oversight. At most 6 singers were on payroll at a given time. Managing them required about 1.5 hours of time per week.

4.2.2 The Resulting Sung Examples

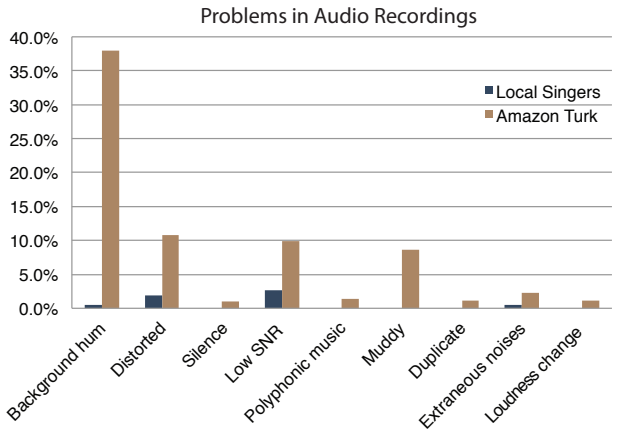

Figure 1. Problems in audio recordings of paid local singers and M. Turk workers.

The paid local singers generated at least 6 sung examples for each of the 100 songs in our test set (see Section 5.1). The mean duration of a single example using this method was 24.46 (SD 6.01) seconds. The examples almost always contain the chorus, the verse or both. A member of our lab listened to the original audio for each example and marked down any audio problems noted. The lab member also performed this task for the recordings by the M. Turk workers as described in Section 4.3.2. The problems for both sets were grouped into a number of natural categories. Out of the examples by the paid local singers, 92% had no problems. The problems of the remaining examples are illustrated in Figure 1. Note that an example can have multiple problems. The problem categories are defined as follows:

background hum The recording contained a noticeable, constant, pitched tone (e.g. 60 Hz AC hum).

distorted The recording was perceptibly distorted.

silence The recording contained only silence.

low SNR The recording seemed to have a low signal-to-noise ratio (SNR) – possibly due to the singing being too quiet.

polyphonic music The recording consisted of polyphonic music (e.g. the commercial recording of the song) instead of an *a cappella* sung example.

muddy The recording was not clear and seemed a bit muffled.

extraneous noises The recording contained significantly noticeable background noises (e.g. kids playing, etc.)

loudness change The recording had a drastic change in loudness in it.

4.3 Using Amazon Mechanical Turk

Amazon's Mechanical Turk service is a micro-task labor market designed for crowdsourcing small tasks requiring human intelligence (i.e. tasks that computers are not yet able to perform well). "Requesters" can post "Human Intelligence Tasks" (HITs) for work that they need done, and "workers" can search through and perform tasks for typically small amounts of money. Amazon acts as the intermediary between the requesters and the workers.

Using the Mechanical Turk service, we crowdsourced the human computation of exemplars for the generation of QBH search keys. We solicited singers by posting a $0.10 HIT entitled: "Sing Popular Songs" and description: "Choose from a variety of genres and songs, and sing the song into your computer". This pay is typical for a task of this difficulty. The workers were given a list of songs to chose from and the Tunebot web application for recording themselves with their computer, which uploaded the recorded content to our server. They were instructed to "sing the most memorable portion of the song (often the verse or melody)". As this is a rather involved task compared to most Mechanical Turk HITs, we tried to make the instructions and interaction as simple and painless as possible. By doing so, we eliminated some of the extra control we had with the paid local singers (e.g. control over who sang which songs, even lyrics/no lyrics ratios, even male/female ratios, vetted singers, etc.).

4.3.1 Computation Time and Cost

The mean time between worker's contributions in a single session was 4.09 (SD 4.78) minutes. In total it took 20 days to collect the 600 sung examples for the 100 songs in the song set. Therefore the M. Turk workers generated about 210 examples per week. When seeking 6 examples per song, the computation time is 24.5 minutes per song. At a pay rate of $0.10 per contributed search key, each song cost $0.60.

4.3.2 The Resulting Sung Examples

While we had 211 unique workers begin our HIT, only 70 of them actually submitted a contribution. 70 unique M. Turk workers contributed the 600 sung examples for the 100 song set. Upon listening to the examples by the 70 workers, we estimate that 25% are male, and 75% are female. The mean number of total examples contributed by a single worker was 8.83 (SD 9.78) and the maximum number of examples contributed by a single worker was 69. Each worker could only contribute 1 example per song. The mean duration of a single example was 26.02 (SD 6.96) seconds and typically consisted of either the verse or chorus.

As with the paid local singers, a lab member listened to each of the 600 M. Turk contributed examples and categorized the problems found. The resulting problems are shown in Figure 1. Only 278 out of 600, or 46%, of M. Turk contribute examples were problem-free. This compares to the 92% problem-free examples for the paid local singers. As illustrated in Figure 1, of the remaining keys, the most common problems were background hum, distortion, 228 had an audible background hum (38% of contributions). Given this, we took steps to minimize the background hum and noise. All M. Turk contributions were processed with a noise reduction algorithm before being converted into search keys. In addition to hum or background noise, a number of the contributed examples did not contain *a cappella* singing. They instead contained silence or polyphonic music.

5. EXPERIMENT 1

5.1 Design

As can be seen in Figure 1, the audio quality of the examples contributed by the paid local singers is higher than the audio quality of the examples contributed by the M. Turk workers. In addition, since the paid local singers were auditioned and ensured to be high quality singers, it is likely that the singing quality is generally superior to that of the M. Turk workers. The question that remains however is whether or not these higher quality audio examples translate to better search keys?

We developed a simple approach to measure the effectiveness of the search keys created with the two human computation methods. For each method, we generated search keys from the sung examples as described in Section 3.1 and inserted these search keys into an existing database of songs used by Tunebot. We then took a set of sung queries drawn either from the M. Turk workers or from the local singers and queried the database. The search key generation method that yielded better search rankings was deemed better.

A run is one presentation of one query per target song (100 queries, total) to the search engine. For each run, we chose a source of search keys (M. Turk or paid local singers) and a source of queries (M. Turk or paid local singers). For each run, we then constructed a database containing all 600 search keys from the chosen source plus approximately 13000 additional distracter search keys from the on-line Tunebot database.

For each of the 100 target songs in the run we performed the following steps:

1. Randomly choose one of the target song's 6+ search keys to be the query and remove this from the set of search keys
2. Remove any other search keys for the target song that are by the same singer as the query (only relevant for the paid local singers).
3. If there are more than 5 remaining search keys in the database, randomly pick 5 of them to remain.
4. Query the database with the chosen query.

The reasoning behind some of these steps is as follows:

- For step 2, recall that a unique worker generated each of the search keys for a particular target song in the M. Turk set. However, in the paid local singer

set this is not the case, and sometimes contributions from the same user for the same song were very similar. Thus, the run in which paid local singers were used for both the query set and search key database, there were sometimes fewer than 5 examples in the database (the mean was 4.52 (SD 0.82)).

- For step 3, note that for the run in which the search keys and queries were both generated by the M. Turk workers, every example that was used as a query was excluded from the set of target search keys placed in the database. Thus, only 5 (instead of 6) examples were available per song as search keys in the database. To maintain consistency across runs, we set the maximum number of examples per song in the database to 5.

We then performed 10 runs (1000 total queries) for each of the 4 combinations of search key and query set sources.

5.2 Results

The measure we used to evaluate the quality of search results is the mean reciprocal rank (MRR) [3]. It is computed as follows:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \quad (1)$$

where $rank_i$ is the rank of the correct target in the search results of the i^{th} query, and N is the total number of queries performed. This measure ranges from 0 to 1, with higher values signifying better search results.

The MRR of the search result matches are reported in Table 1. When using paid local singer contributions as queries, the performance of the runs with paid local singer generated search keys was better than the performance of the runs with the M. Turk generated search keys ($p = 0.000014$ using a Wilcoxon signed rank test). However, when using the M. Turk contributions as queries, there was not a statistically significant difference between the performance of the runs with paid local singer generated search keys and runs with M. Turk generated search keys ($p = 0.29$ using a Wilcoxon signed rank test). Therefore, while it seems that the search keys generated by the paid local singers are of higher overall quality, this only affects the search results when they are also queried by high quality search keys. It seems reasonable to assume though that real-world queries are likely more similar to the M. Turk contributions than they are to paid local singer contributions. If that is the case, then it is not worth the higher cost of the local singers in order to have more control over the process and higher quality search keys.

6. EXPERIMENT 2

6.1 Design

In the previous experiment we limited the databases to 5 search keys per song. This however leads us to the following 2 questions:

1. Does the performance of the system improve with additional search keys?

Target Source	Query Source	Mean Reciprocal Rank (95% CI)
M. Turk	M. Turk	0.2140 [0.1909,0.2367]
L. Singers	M. Turk	0.2453 [0.2229,0.2705]
M. Turk	L. Singers	0.2942 [0.2710,0.3218]
L. Singers	**L. Singers**	**0.3781 [0.3467,0.4047]**

Table 1. Mean reciprocal rank (MRR) of the search results of 1000 queries for each of the 4 source / query set combinations. MRR ranges from 0 to 1, with higher values signifying better search results.

2. How many search keys are needed to reach an of MRR of 0.5 (a performance level in which the target is in the top couple of results at least half the time)?

To answer these questions, we designed another experiment. For this experiment, we utilized the full dataset (including all of the search keys that were originally distracters), observed how the performance increased as the number of search keys increased, and noted at what point the performance achieved an MRR of 0.5.

Since the search performances of paid local singer generated search keys and M. Turk generated search keys were not significantly different when queried by the M. Turk queries, we utilized the full dataset as the set of evaluation search keys. This set included search keys from the paid local singers, the M. Turk workers, and all user contributions (the "citizen scientist" market). This dataset contains a total of 2587 songs with 14509 search keys. The number of search keys per song ranges from 1 to 61. Chance performance (i.e. random search results) on this dataset would be an expected MRR of 0.00326. We used all of the search keys except those by the paid local singers when creating our set of queries. By doing so we made the assumption that all of the crowdsourced keys – those created by user contributions ("citizen singers") and M. Turk workers were of similar quality. This was a necessary assumption to utilize the larger dataset. To generate the search results, we then queried the search keys with each of the queries. As in Experiment 1, each time we queried we removed all of the search keys that were also of the same musical work by the same singer.

6.2 Results

To analyze the search results, we grouped the rankings into sets by target songs which all had equal numbers of search keys, e.g. all of the rankings for songs with 8 search keys were grouped together, etc. We took the means of each of those sets as shown in Figure 2a. The total number of search keys in each of these sets is shown in Figure 2b. We then transformed the means using an exponential to account for the curvature of the data, and then fit a line using ordinary least squares linear regression. This line is the brown dotted curve in Figure 2a. Examining Figure 2a, we find the MRR does improve as we add additional search keys (thereby answering Question 1), and it does not seem to converge within the range that we ex-

	Paid Local Singers	M. Turk Workers
Mean duration of search keys (sec)	24.46	26.02
Cost ($ per search key)	0.64	0.10
Cost – 0.5 MRR ($ per song)	11.52	1.80
Computation time (min per search key)	4.26 (SD 3.79)	4.09 (SD 4.78)
Time to generate 100 song set (days)	9.5	20
Approximate oversight time (hours / week)	1.5	1

Table 2. Comparison of paid local singers and M. Turk workers

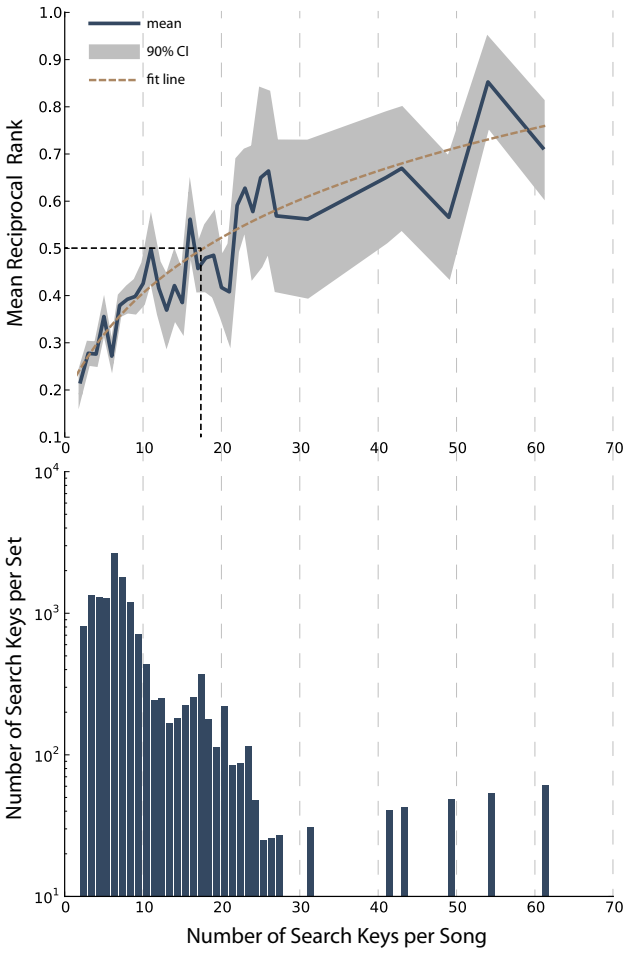

Figure 2. TOP (a): The mean reciprocal rank as a function of the number of search keys per song as described in Section 6. The grey area is the 90% confidence interval around the mean. The brown dotted line is the regression fit. BOTTOM (b): The number of search keys used when calculating the means in the top figure. Note than there were far fewer total search keys used when calculating the means on the right side of the graph.

amined. We also find that if we are seeking search performance with an MRR of 0.3781 (the performance of the paid local singer query and search keys in Experiment 1), we would need 10 examples, and if we were seeking performance with a MRR of 0.5, we need to acquire about 18 search keys per song (thereby answering Question 2). With this knowledge, we calculated the cost per song for search performance of MRR 0.5 and found that for M. Turk workers it was $1.80 and for paid local singers it was $11.52.

7. CONCLUSIONS

In this paper we compared two methods of human computation for generating sung search keys for a query-by-humming search engine: hiring paid local singers and crowdsourcing Amazon's Mechanical Turk (M. Turk) workers. We showed the system works best when paid local singers are used to generate both the search keys and the queries. It's likely however that the M. Turk generated queries are more like real world queries, and in the scenario in which M. Turk contributions were used as the queries, there was no significant difference in performance between the search key databases generated by the two human computation methods. Considering that the crowdsourced, M. Turk generated search keys cost 15.62% of the cost of the paid local singers, this seems like a promising approach. We also showed that the mean reciprocal rank (MRR) of the QBH search engine will improve when more search keys are added to the database, and that roughly 18 search keys are required to achieve a MRR of 0.5.

It however took roughly twice as long to generate the keys for the 100 target songs using the crowdsourced method when paying a rate of $0.10 per search key. This was not due to the human computation time, which was about the same for both human computation methods, but rather the *discovery time* [12] of the HIT. While other work in the field of human computation [20] has showed that this time can be reduced by paying higher rewards per task, increasing the reward may also increase the number of invalid search keys generated by workers trying to cheat the system. Therefore, in future work, we will both work to find the optimal price to achieve our desired completion time as well as develop a method to vet the search keys as they are contributed.

Acknowledgments

The authors would like to thank Gabriel Peal for his help with the data collection.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0824162, and in part by National Science Foundation Grant No. IIS-0812314.

8. REFERENCES

[1] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. of International Symposium on Music Information Retrieval*, Paris, France, 2002.

[2] A. Wang, "An industrial strength audio search algorithm," in *Proc. of Fourth International Conference on Music Information Retrieval*, Baltimore, 2003.

[3] R. B. Dannenberg, W. P. Birmingham, G. Tzanetakis, C. Meek, N. Hu, and B. Pardo, "The musart testbed for query-by-humming evaluation," *Computer Music Journal*, vol. 28, no. 2, pp. 34–48, 2004.

[4] N. Hu, R. B. Dannenberg, and A. L. Lewis, "A probabilistic model of melodic similarity," in *Proc. of International Computer Music Conference*. San Francisco: International Computer Music Association, 2002.

[5] R. Typke, P. Giannopoulos, R. Veltkamp, F. Wiering, and R. Van Oostrum, "Using transportation distances for measuring melodic similarity," in *Proc. of International Conference on Music Information Retrieval*. Citeseer, 2003, pp. 107–114.

[6] R. Typke, F. Wiering, and R. Veltkamp, "A search method for notated polyphonic music with pitch and tempo fluctuations," in *Proc. of International Conference on Music Information Retrieval*. Citeseer, 2004, pp. 281–288.

[7] M. Cartwright, Z. Rafii, J. Han, and B. Pardo, "Making searchable melodies: Human vs. machine," in *Proc. of 3rd Human Computation Workshop (HCOMP 2011)*. San Francisco: AAAI Press, 2011.

[8] SoundHound, "Soundhound inc." 2012. [Online]. Available: http://www.soundhound.com

[9] A. Huq, M. Cartwright, and B. Pardo, "Crowdsourcing a real-world on-line query by humming system," in *Proc. of 7th South and Music Computing Conference (SMC 2010)*, Barcelona, 2010.

[10] Z. Rafii and B. Pardo, "A simple music/voice separation method based on the extraction of the repeating musical structure," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011)*, Prague, Czech Republic, 2011.

[11] J. Han and C.-W. Chen, "Improving melody extraction using probabilistic latent component analysis," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011)*, Prague, Czech Republic, 2011.

[12] E. Law and L. v. Ahn, "Human computation," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 5, no. 3, pp. 1–121, 2011.

[13] L. v. Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. of SIGCHI conference on Human factors in computing systems*. Vienna, Austria: ACM, 2004.

[14] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg, "Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey," *Monthly Notices of the Royal Astronomical Society*, vol. 389, no. 3, pp. 1179–1189, 2008.

[15] Duolingo, "Duolingo," 2012. [Online]. Available: http://duolingo.com/

[16] D. A. Shamma and B. Pardo, "Karaoke callout: using social and collaborative cell phone networking for new entertainment modalities and data collection," in *Proc. of 1st ACM workshop on Audio and music computing multimedia*. Santa Barbara, California, USA: ACM, 2006.

[17] B. Pardo and D. Shamma, "Teaching a music search engine through play," in *Proc. of CHI 2007 Workshop on Vocal Interaction in Assistive Technologies and Games*, San Jose, CA, USA, 2007.

[18] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proc. of Institute of Phonetic Sciences*, vol. 17, 1993, pp. 97–110.

[19] R. Durbin, *Biological sequence analysis : probabalistic models of proteins and nucleic acids*. Cambridge, UK New York: Cambridge University Press, 1998.

[20] S. Faridani, B. Hartmann, and P. Ipeirotis, "What's the right price? pricing tasks for finishing on time," in *Proc. of 3rd Human Computation Workshop (HCOMP 2011)*. San Francisco: AAAI Press, 2011.