A Probabilistic Ranking Model for Audio Stream Retrieval

YoungHoon Jung
Dept. of Computer Science
Columbia University
New York, NY 10027
jung@cs.columbia.edu

Jaehwan Koo
Dept. of Information Solutions
I-Yuno Media Group
Burbank, CA 91502
cto@iyunomg.com

Karl Stratos
Dept. of Computer Science
Columbia University
New York, NY 10027
stratos@cs.columbia.edu

Luca P. Carloni
Dept. of Computer Science
Columbia University
New York, NY 10027
luca@cs.columbia.edu

ABSTRACT

In Audio Stream Retrieval (ASR) systems, clients periodically query an audio database with an audio segment taken from the input audio stream to keep track of the flow of the stream in the original content sources or to compare two differently edited streams. We recently developed a series of ASR applications such as broadcast monitoring systems, automatic caption fetching systems, and automatic media edit tracking systems. Based on this experience, we propose a probabilistic ranking model designed for ASR systems. In order to train and test the model, we create a new set of audio streams and make it publicly available. Our experiments with these new streams confirm that the proposed ranking model works effectively with the retrieved results and reduces the errors when used in various ASR applications.

1. INTRODUCTION

Audio segment retrieval, i.e. searching information about the original audio content with a query of an audio segment, is an emerging technology in the area of audio information retrieval [9, 22]. This can be viewed as a special type of information retrieval, called *Query by Example* (QbE), that searches results identical or similar to the example provided in the query from the user instead of searching with the constraints or keyword terms in the query [21, 23]. Likewise, in audio segment retrieval the user's submitted query contains an audio segment as an example and the retrieval server returns information about identical or similar audio sources. One of the most widely used applications of audio segment retrieval is music identification [1, 7, 14], which takes a segment of music to search the information on the original music from the database server. While the input query in audio segment retrieval is mostly a piece of audio, a typical output result may consists of multiple items, including: 1) metadata such as the creator, the date of creation, the ID, or the title of the content [4]; 2) means to access to the whole content (or similar content [5]); 3) derivative works like a subtitle, a caption, or a lyric file [8]; and 4) the relation between the input segment and the original content, i.e. the position of the input segment in the searched content [11].

We have developed a new class of applications where the action of audio retrieval is performed online with queries taken from an audio stream. In particular, we developed a real-time broadcast monitoring system that allows stations to monitor the stability of the actual broadcast and allows advertisers to count how many times their commercials are exposed on air. We also developed an automatic caption fetching system that samples an audio segment from a video stream played by the user and uses it to determine the proper caption to display together with the video. We call this new class of applications *Audio Stream Retrieval* (ASR) as it has certain characteristics distinct from traditional audio retrieval. First, in ASR the result usually consists of a pair of items: a content ID and the position of the audio in the query within the retrieved result; this allows ASR applications to track the sequential progression of the input stream. Second, ASR involves multiple, periodic, and online retrieval actions over the sequential audio stream. ASR requires both high precision and high recall to ensure that the relevant result from the database is consistently included in the result set. With the goal of meeting these requirements, in this paper we propose a probabilistic ranking model for ASR. We evaluate the proposed model with a set of experiments that confirm its effectiveness. For these experiments, we developed a new suite of audio streams for training and testing that we have made available in the public domain.

2. AUDIO STREAM RETRIEVAL

In this section, we explain some concepts and background information necessary to understand the development of the ranking model for ASR.

2.1 Audio Streams

A *content source* or simply *content* is the original audio or video file from which the fingerprints for database is generated. An audio or video *stream* is a sequence of excerpts from various content sources. ASR clients create fingerprints from the streams, combine them into a query with the fingerprints, and submit the query to the retrieval server.

Fig. 1 shows two examples on how streams are made of multiple contents. The figure presents also queries q_k and the ideal result r_k corresponding to q_k. The top example is a typical stream that can be observed on TV channels. During a TV show (Content 0) some commercials (Content 1 and 2) are played intermittently in the middle of the show. The other example presents a user who watches different content sources back-to-back. The user may watch the content from the beginning (or any arbitrary position) to the end (or any other arbitrary position). Across these two examples, the

MARMI'16, June 06, 2016, New York, NY, USA.
© 2016 ACM. ISBN 978-1-4503-4362-6/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2927006.2927013

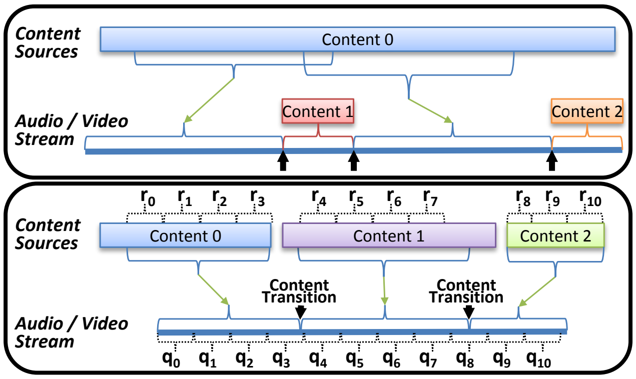

Fig. 1: Two examples of audio streams: an edited stream for a TV channel with commercials (top) and a stream with a user occasionally changing the content source (bottom).

goal of ASR is to find out the original content sources and the positions from which the excerpts came.

Content transition, or simply *transition*, is a change of the content source in a stream. The point of transition in the streams of Fig. 1 are indicated with black arrows.

2.2 The Query

The ASR systems introduced in this paper are based on audio fingerprints [6, 16]. A query consists of fingerprints created from the input stream [2, 13]. In ASR, the client makes a query after every *query interval* $\Delta t = t_{k+1} - t_k$. The k-th query takes an audio segment from t_k to t_{k+1} of the audio stream. The time notation means content time, which is not necessarily real-time. The query interval is real-time in the automatic caption-fetching system because it also plays the video or audio. Instead, in the automatic media edit tracking system that compares two audio streams without playing them, the client sends another query as soon as the results from the previous query return.

The *query length* λ is the length of the audio segment used to make the query. The query length is a concept that is independent from the query interval. A longer query length may allow the retrieval server to pick up more precise results out of a large number of similar content sources, but it may also cause confusion for the same server if the content in the input stream changes very frequently (shorter than the query length). Meanwhile, a shorter query interval may lead to a faster discovery of content changes but it can burden the retrieval server with too many requests, thus limiting its scalability. Therefore, the independence of these two concepts, query length and query interval, allows more configuration flexibility in the ASR systems.

2.3 The ASR Workflow

Fig. 2 shows the flowchart of the proposed audio stream retrieval. It is an iterative process in which the client and the retrieval server interact repeatedly. The client decodes the audio or video content stream, extracts an audio buffer, and generates fingerprints. Then, it creates a query that includes the generated fingerprints. If it is not the very first query in the stream and the client has previous results, then these previous results are also included in the query to be used in the retrieval process, e.g. result ranking.

There are two ways to use information from the previous retrieval results. First, the server can maintain the previous results in a session-based retrieval system. Second, the client can keep the previous results and send a query with the previous results in a session-less system. In our systems, we use the latter type for the design of queries as illustrated in the flowchart.

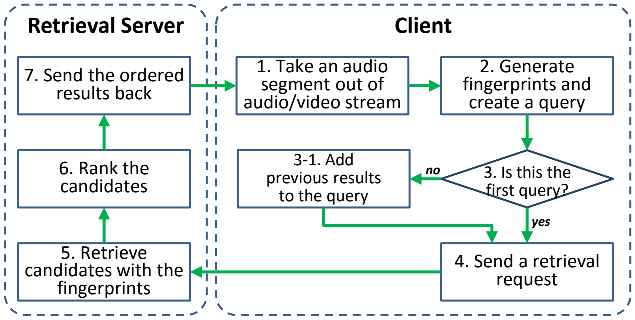
Retrieval Server
Client
7. Send the ordered results back
1. Take an audio segment out of audio/video stream
2. Generate fingerprints and create a query
6. Rank the candidates
3-1. Add previous results to the query
no
3. Is this the first query?
yes
5. Retrieve candidates with the fingerprints
4. Send a retrieval request

Fig. 2: Flowchart of the proposed ASR.

The server searches its database with the last fingerprint in the query and gives a ranking score for each retrieved result. These results are sent back to the client. Many text-based information retrieval systems use the ranking model not only to rank the documents but also to retrieve documents from the database. Differently from these text-based retrieval systems, the ASR systems use separate models for retrieval and ranking. In this paper we focus on the ranking of the ASR systems.

3. THE PROPOSED RANKING MODEL

In information retrieval, ranking is a way to provide the user with an estimate on the relevance of documents returned in response to a given query. Since typically a user cannot check all the retrieved documents or is interested only in the most relevant document [12], a higher ranking suggests that a given document may be very relevant. Ranking the results is important in audio stream retrieval for the same reasons. In addition, the results of ASR can be used as a part of automated systems in many applications.[1] For example, the broadcast monitoring system uses only the highest ranked result and the automatic caption display system employs a client-side heuristic-based module that picks one of the top-k results.

One approach used in ranking audio retrieval can be used also as a basic method to rank audio stream retrieval results: i.e. to use the distance between the fingerprints in the query and the fingerprints in the retrieved content sources. The rank achieved by the distance works well as long as there are no content sources that have similar portions. If there exist, however, some content sources that have similar portions, then it is important to distinguish the results because their distances to the query will be too short to be excluded. The more relevant result should be the actual content and position of the audio the user is feeding, the truth value of which, however, we cannot know. The less relevant results should be the content sources and the positions of audio segments that are different from the truth value but have a similar portion to what the user is feeding. This is a problem almost im-

[1] There are also applications, e.g. copyright monitoring systems, where the users want to identify all the streams that use a certain portion of their work. In this case, high recall is more important than ranking.

possible to solve by using only fingerprint distances because there are content sources that contain precisely identical audio parts, e.g. a same song or a sound effect used in two different movies.

Wherever the truth values of the relevance property of each result is unknown to an information system, we assume that the information available for ranking in the system is at best probabilistic. This is motivated by the Probability Ranking Principle [12]. Next, we present a probabilistic ranking model for audio stream retrieval based on the given information in the database and the query. In audio stream retrieval, we can probabilistically anticipate the current results based on the knowledge of the previous results. The analysis of our applications in production makes us observe the presence of a probabilistic relation that characterizes the transition between content sources. This is due to the following reasons: 1) today's video-content hosting services provide their users with recommendations which often lead to a high correlation in the transitions between similar videos; 2) the users have a particular interest that may affect their watch patterns, e.g. watching all the soccer games a team has played during a league or watching a sequence of episodes of a TV show; and 3) a publicly available video stream (or channel) is watched by many users, i.e. many users watch the same sequence of video content sources such as TV commercials and shows, which the video stream (or channel) contains.

3.1 Notation

For the development of an ASR ranking model, we borrow and extend the following notation from the Probabilistic Relevance Framework [18]. The binary relevance between the query and the result is represented as either rel (relevant) or $\overline{\text{rel}}$ (not relevant). Two monotonic functions f_1 and f_2 are *equivalent as ranking functions* ($f_1() \propto_q f_2()$). A result $\mathbf{r}$ is a pair of a content ID c and a position in the content p. C is a set of the IDs of all the possible content sources in the system. Querying the database returns a set $\mathbf{R}$ of results ordered by their ranking scores. As mentioned in Section 2.3, an audio stream retrieval query comprises also the results of the last query. Hence, we define a query $\mathbf{q}$ as a combination of a list of fingerprints and a previous top-ranked result. We denote the k-th query as:

$$\mathbf{q}_k = (\{\text{FP[i]} \mid 0 \leq i < n \text{ and } \text{FP}[i] \in \mathcal{F}\}, \mathbf{R}_{k-1}.r_0)$$

where $\mathbf{R}_k$ denotes the retrieved results for q_k and $\mathcal{F}$ denotes the set of all possible fingerprints in the feature space.

3.2 Ranking Model Development

We estimate the relevance between a query and a result using three factors: ① the history (or the previous result) of how the content sources in the input stream are changing; ② the distance between the fingerprints in the query and the ones the result indicates; and ③ the heuristics that in many retrieval cases the duration of the excerpted content portions in the streams is usually longer than the query interval. In particular, ② is what can be used also for audio retrieval whereas ① and ③ are information available only in ASR.

We evaluate each result $\mathbf{r}$ in $\mathbf{R}_k$ retrieved from the database with the last fingerprint in the query $\mathbf{q}_k$ as shown in Fig. 2. The relevance of a result $\mathbf{r}$, a pair of content c and position p to a query $\mathbf{q}$ can be expressed by $P(\text{rel}|c,p,\mathbf{q})$. We use the relevance probability as the ranking score so that the results are sorted by this score in descending order. We transform the relevance probability with the following steps:

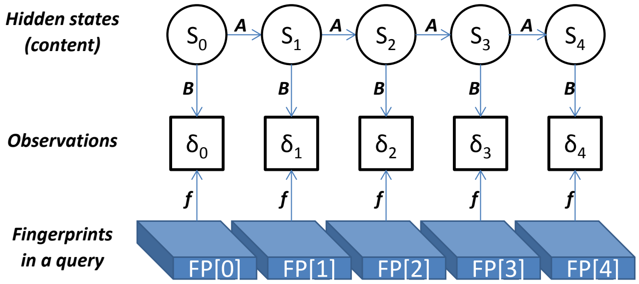

Fig. 3: Content Probability based on HMM.

$$P(\text{rel}|c,p,\mathbf{q}) \propto_{\mathbf{q}} \frac{P(\text{rel}|c,p,\mathbf{q})}{P(\overline{\text{rel}}|c,p,\mathbf{q})} \tag{1}$$

$$= \frac{P(c,p|\text{rel},\mathbf{q})}{P(c,p|\overline{\text{rel}},\mathbf{q})} \frac{P(\text{rel}|\mathbf{q})}{P(\overline{\text{rel}}|\mathbf{q})} \tag{2}$$

$$\propto_{\mathbf{q}} \frac{P(c,p|\text{rel},\mathbf{q})}{P(c,p|\overline{\text{rel}},\mathbf{q})} \tag{3}$$

$$= \frac{P(c|\text{rel},\mathbf{q}) \cdot P(p|\text{rel},c,\mathbf{q})}{P(c,p|\overline{\text{rel}},\mathbf{q})} \tag{4}$$

First, the relevance probability is transformed by odds-ratio in Equation (1). We get Equation (2) by applying Bayesian inversion. In Equation (3) we drop the terms that are not related to the results (c and p), while preserving the ranking. Finally, the probabilities of content and the position are split in Equation (4).

Content Probability. The term $P(c|\text{rel},\mathbf{q})$ is the probability of c given the query $\mathbf{q}$ and a relevance rel of a result $\mathbf{r}$ (c, p) with respect to $\mathbf{q}$. To estimate this probability we apply the Hidden Markov Model (HMM) [3] to ASR by treating the content sources as the hidden states and the fingerprints as the observable outputs. Using this model, which is illustrated in Fig. 3, we can compute the probability that c is the current content given the fingerprint sequence in the query. We assume that the probability of a change between content sources is affected only by the current content, not the previous ones. This allows us to use HMM for estimating the content probability.

On top of HMM, we use the Forward algorithm [20] to calculate a 'belief state', the probability of a state at a certain time, given the history of observations. For instance, in Fig. 3, we can get the probability of each content source in the retrieved results being the current content c in the stream, i.e. $S_4 = c$. Thus, the result of the Forward algorithm for c on the last hidden state is the probability of c being the current content from which the last fingerprint FP[4] has been generated.

One of the advantages of HMM is that the observation and state sequence length can vary. The number of fingerprints in the query can also vary due to the different offsets and the frequencies of the query interval and the audio stream. For instance, one query may have 40 fingerprints while the next query has 42 fingerprints, although their buffer lengths are set to be the same. Thus, HMM is a good method to compute the content probability using the fingerprint sequence.

There are two unknown probabilities sets in HMM. The state transition probability $A = P(c_i|c_{i-1})$ denotes the likelihood that the content is changed from c_{i-1} to c_i. The emission probability $B = P(\delta_i|c_i)$ denotes the likelihood that δ_i can be observed when the content is c_i. In ASR, A implies the probabilities of a transition between content sources and B implies the probabilities of a fingerprint being observed at a state. These HMM probability parameters are learned as described later in Section 3.3.

If the size of the states and the observation space are large then the amount of computation required for training the HMM parameters becomes high. For this reason, we reduce the number of states and the observation space in our production systems, e.g. by clustering the content sources and picking the top-k content sources. Also, since the fingerprint space $\mathcal{F}$ is too large to efficiently compute the parameters, we reduce it so that it maps into a smaller observation space by using a dimension-reduction algorithm such as Vector Quantization [10]. An observation value δ is obtained from a fingerprint by a dimension-reduction function f:

$$\delta_i = f(FP[i]). \tag{5}$$

where i is the index of the fingerprints in the query.

Thus, the probability that content c is relevant to a given query $\mathbf{q}$ is calculated by the Forward algorithm which computes the probability $\alpha_t(x)$ that the t-th hidden state is x:

$$\begin{aligned} P(c_i|\text{rel}, q) &\propto_{\mathbf{q}} \alpha_i(c_i) \\ &= P(c_i, \delta_{1:i}) \\ &= \sum_{c_{i-1}} P(\delta_i|c_i, c_{i-1}, \delta_{1:i-1}) \cdot P(c_i|c_{i-1}, \delta_{1:i-1}) \\ &\qquad \times P(c_{i-1}, \delta_{1:i-1}) \\ &= P(\delta_i|c_i) \sum_{c_{i-1}} P(c_i|c_{i-1}) \cdot \alpha_{i-1}(c_{i-1}) \end{aligned} \tag{6}$$

where c_i is the content, or the hidden state, from which the i-th fingerprint in the query has been created.

Therefore, we get an expanded ranking formula:

$$P(\text{rel}|c, p, \mathbf{q}) \propto_q \frac{\alpha(c) \cdot P(p|\text{rel}, c, \mathbf{q})}{P(c, p|\overline{\text{rel}}, \mathbf{q})} \tag{7}$$

by replacing the content probability with $\alpha(c)$, the probability that the last hidden state is the content c given the query $\mathbf{q}$.

Result Probability. The probability of the position p, given the content c, the query $\mathbf{q}$, and the relevance rel between the result and the query can be computed by the distance of the fingerprints. In other words, if the distance between the two fingerprints from the result and the query is big, it is unlikely that the result is relevant to the query. The average distance between two fingerprint sequences X and Y over the k last fingerprints is:

$$d(X, Y, k) = \frac{\sum_{i=n-k-1}^{n-1} ||X_i - Y_i||}{k} \tag{8}$$

where $||x - y||$ denotes the Euclidean distance between two fingerprints x and y and $k \in [2, n)$ is the number of compared fingerprints in the sequence. A weighted and inverted distance is:

$$d^I(X, Y) = \max_k \left\{ \frac{1}{1 + d(X, Y, k)} \cdot \frac{\Gamma(\lfloor k+1 \rfloor, 4)}{\lfloor k \rfloor !} \right\} \tag{9}$$

where $\frac{\Gamma(\lfloor k+1 \rfloor, 4)}{\lfloor k \rfloor !}$ is the weight term based on a cumulative distribution function of the Poisson distribution. This function gives more credibility to the distance values obtained from longer sequences while it reaches a plateau at a certain length of the sequences.

The probability of the position p given c, rel, and $\mathbf{q}$ from the distance function is:

$$P(p|\text{rel}, c, \mathbf{q}) \propto_q d^I(\mathbf{q}.\text{FP}, \text{FP}_{\mathbf{r}}) \tag{10}$$

where $\mathbf{q}$.FP is the fingerprint sequence in the query and $\text{FP}_{\mathbf{r}}$ is the sequence obtained with the last fingerprint in the result $\mathbf{r}$ using the hybrid approach.

The expanded ranking formula is:

$$P(\text{rel}|c, p, \mathbf{q}) \propto_q \frac{\alpha(c) \cdot d^I(\mathbf{q}.\text{FP}, \text{FP}_{\mathbf{r}})}{P(c, p|\overline{\text{rel}}, \mathbf{q})} \tag{11}$$

Irrelevant Result Probability. Let $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$ be two adjacent queries with a query interval Δt. Suppose that r_{k-1} is the top-ranked result retrieved by $\mathbf{q}_{k-1}$. If one result r_k retrieved by $\mathbf{q}_k$ has the same content as r_{k-1} and its position is Δt behind the position of r_{k-1}, then it is likely that there were no content transitions between the two queries in the stream. Therefore, this result is less likely to be irrelevant. For example, if $\text{R}_k.\mathbf{r}_0 = (0, 2000)$ and $\text{R}_{k+1}.\mathbf{r}_0 = (0, 3000)$ while the predefined source interval time Δt is 1000, we know that $\text{R}_{k+1}.\mathbf{r}_0$ would be the correct result with a high probability. Then, these results $\text{R}_k.\mathbf{r}_0$ and $\text{R}_{k+1}.\mathbf{r}_0$ are said to be *contiguous*. We use this heuristic information to model the probability U of a result when the query and the result are irrelevant for a given $\mathbf{q}$.

$$\begin{aligned} U &= \begin{cases} 0, & \text{if } r.c = q.c \text{ and } q.p + \Delta t = r.p \\ \frac{1}{|R|}, & \text{otherwise} \end{cases} \\ &\propto_q (U + \frac{1}{|R|}) \cdot \frac{|R|}{2} \\ &= \begin{cases} 0.5, & \text{if } r.c = q.c \text{ and } q.p + \Delta t = r.p \\ 1.0, & \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

Due to the various different sampling and fingerprinting intervals, the position within the content always contains some small errors, thus making the comparison impractical. We revise this function with a probabilistic normal distribution function:

$$\mathbb{U} \approx 1 - [r.c = q.c] \cdot 1.25 \cdot \mathcal{N}(x, \mu, \sigma^2) \tag{13}$$

where [P] is an Iverson bracket which returns 1 if P is true, $x = 4\frac{r.p}{\Delta t}$, $\mu = 4\frac{q.p}{\Delta t}$, and $\sigma^2 = 1.0$. This results in the final ranking formula:

$$P(\text{rel}|c, p, \mathbf{q}) \propto_q \frac{\alpha(c) \cdot d^I(\mathbf{q}.\text{FP}, \text{FP}_{\mathbf{r}})}{\mathbb{U}} \tag{14}$$

This ranking model is used to calculate a ranking score for each result. The retrieval server sorts the results with their ranking scores in descending order.

3.3 Learning Parameter Values

The two HMM parameters we used in the proposed ranking model are one of the key factors to make it successful. The ideal values for these parameters are different for various applications. Even with the same applications, different composition of content sources in the database or different user-transition patterns observed in the input streams require different parameter values for the best ranking performance. For this reason, many recent ranking systems learn the parameter values used in the ranking model to achieve more appropriate parameter values, thus satisfying their users better.

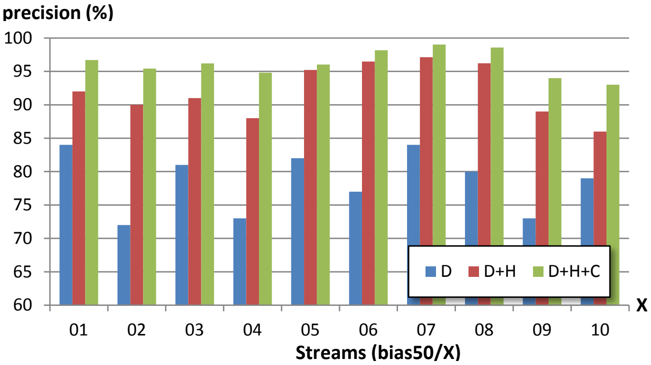

Fig. 4: Precision of top-ranked result.

This is called *Learning to Rank* [15]. By applying machine learning algorithms the ranking model can work more closely to the user patterns.

When applying machine learning, it is difficult under certain circumstances to obtain the training dataset for the learning model. We collect training datasets from the ASR applications that include the input (a query) and labeled output (a result) for the ranking model. We extracted from the ranking model datasets the input (dimension-reduced fingerprints and the previous content) and output (a content source) to train and test the embedded HMM. This allows us to train the HMM parameters separately from the ranking model. We obtain the initial emission probabilities from the fingerprint database for each content source by counting the frequencies of fingerprint observations. This accelerates learning of the emission probabilities based on the actual occurrence of fingerprints within the input stream.

We use two different methods to learn the HMM parameters for systems under different circumstances, particularly regarding the availability of the label (truth value) of the current content in the stream for training. First, in those systems where the label of the content in the given training stream is available (supervised training), we calculate the transition probabilities and the emission probabilities online, by using the content label and the observations generated from the fingerprints in the query. Second, in the systems where the content label is unavailable (unsupervised training), we use the Baum-Welch algorithm to learn the HMM parameters. The Baum-Welch algorithm is based on an expectation-maximization algorithm to find the maximum likelihood estimate of the parameters of a HMM given a set of observed feature vectors [17].

4. EXPERIMENTS

For the evaluation of our ranking model, we created *Open Audio Stream*, a suite of audio streams[2] that present distinct characteristics in terms of content transitions. The streams consist of excerpts from some audio field recording archives named freefield1010 [19]. In the streams we created, transitions are biased in the sense that the transitions from a content source tend to converge to a few content sources rather than being evenly spread across many content sources. The stream-group names *bias0* and *bias100* correspond to unbiased (completely random) streams and completely biased streams, respectively. More detailed information is available on the website. The experiments on the Open Audio Stream discussed from Section 4.1 to 4.3 were made with a server that had been loaded with all the fingerprints created from the 7,690 freefield1010 wave files.

Query		D			D+H		
c	p	c	p	Score	c	p	Score
110917	7800	**110917**	**7811**	**0.34**	**110917**	**7811**	**0.75**
		110917	3014	0.07	110917	3014	0.07
110917	7900	*169249*	*9230*	*0.19*	**110917**	**7904**	**0.33**
		110917	**7904**	**0.17**	*169249*	*9230*	*0.19*
		168490	3014	0.15	168490	3014	0.15

Table 1: Ranking Score Example: D vs. DH.

Query		D+H			D+H+C		
c	p	c	p	Score	c	p	Score
170470	9200	**170470**	**9230**	**0.49**	**170470**	**9230**	**0.374**
		159171	3955	0.34	159171	3955	0.012
17077	600	*42200*	*7911*	*0.35*	**17077**	**598**	**0.054**
		162452	*5462*	*0.33*	*42200*	*7911*	*0.011*
		17077	**598**	**0.32**	*162452*	*5462*	*0.009*
		69965	3955	0.15	69965	3955	0.004

Table 2: Ranking Score Example: DH vs. DHC.

4.1 Ranking Model Evaluation

In this section we evaluate the precision of our ASR system expressed as the average percent of correct top-ranked results out of the total number of queries. We included ten sets of streams from the *bias50* directory of the Open Audio Stream. We evaluate the ranking model by comparing the three versions with different levels of sophistication, as there is no existing ranking model for ASR to be used as a baseline. For each set of streams we ranked the retrieved results based on: D (the distance-based result term only), D+H (D and the irrelevant result term using previous results), and D+H+C (D, H, and the content term). As shown in Fig. 4, for each set of streams the more elaborate rank model yields the more precise ranking results. In particular, H improves the precision from 10%points to more than 30%points. Thanks to the additional improvement brought by C, the complete ranking model (D+H+C) results in a precision of over 95% across all eight evaluated streams.

4.2 Evaluation Breakdown By Examples

Here we delve into some examples gathered during the experiments discussed in Section 4.1 to show how the precision improvement was actually achieved. Table 1 shows two queries from the stream *biased10/07/0019*. The portion of the stream from which the two queries are taken is originally made from content source with id *110917* and positions from 7800 ms to 7900 ms and from 7900 ms to 8000 ms, respectively. Retrieving with these queries and ranking should bring the results with the same content and positions as top-ranked ones. For instance, with the D ranking model, the first query (110917@7800) brings two results where the top-ranked one is (110917@7811), which is correct (because the system allows positions within ± 250 ms in this example). The second query, however, brings three results where the supposedly-correct one is ranked second. This incorrect behavior may happen whenever there are very similar parts in the content database. The D+H model, on the other hand, ranks (110917@7904) as the top for the second query while

[2] Open Audio Stream is available at *http://openaudiostream.org*

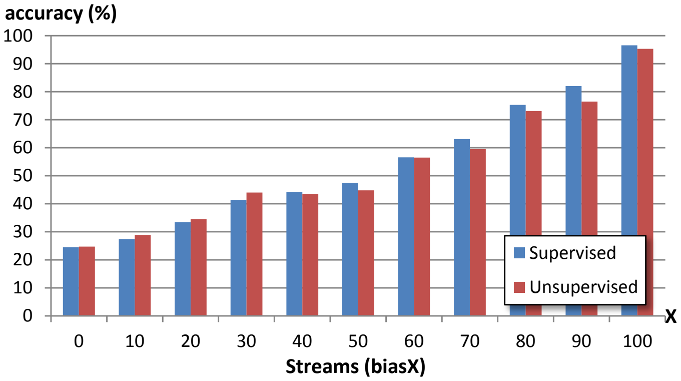

Fig. 5: Content Prediction Accuracy.

keeping the rank of the first query also correct. As the two queries are adjacent, having one query interval $\Delta t = 1$ second in-between, the irrelevant result term boosts the score of the result of the second query up to 2 times. This is one of the most powerful factors of the ranking model as points that can benefit from this score boosting are very frequently observed in the user streams.

Table 2 shows two queries from the stream *bias60/03/0012* and their retrieval results ranked by two models: D+H and D+H+C. The portion of the stream from which the two queries are taken is composed of two different sources: from 9200 ms to 10000 ms of *170470* and from 400 ms to 1400 ms of *17077*. In other words, there is a transition in this part of the stream, changing from *170470* to *17077*. The dataset we used for training and testing the ranking model, particularly the HMM-based content probability, is *bias60* of the Open Audio Stream. With D+H, the first query gets a correct top-ranked result (170470@9230). For the second query, however, the correct result (17077@598) is ranked third. This also happens when there are similar sounds in other content sources. On the other hand, the D+H+C model fixes this problem by multiplying the score with the probability of the content source *17077* being the content of the second query. This is computed with the HMM Forward algorithm using parameters trained on the *bias60* streams and the fingerprints in the query.

4.3 HMM Evaluation

This section shows the accuracy of content prediction, calculated by the number of correctly estimated most-likely content sources. Fig. 5 shows the values obtained with sets of streams from the Open Audio Stream. The ranking model is trained with the 10 streams randomly selected from each set and is tested against all the streams in the group. In the results on the Open Audio Stream datasets (*bias30* - *bias90*), the accuracy increases as the streams are more biased. This is because the state-transition probabilities learned from some streams in a dataset are likely to work well with other streams in the same dataset if the transitions in that dataset are more biased. In many cases, supervised training yields a performance improvement of about 5%points to 10%points.

5. CONCLUSION

We introduced audio stream retrieval (ASR) as a type of audio information retrieval that has some distinct characteristics such as: periodic querying, content excerpt length usually longer than the query interval, and the presence of content transitions in the input streams. For ASR, we proposed a probabilistic ranking model that is based on the distance of fingerprints between the query and the retrieved results. The ranking model uses ASR-specific information to improve the ranking results. We developed a suite of audio streams for training and testing purposes that we made publicly available online. The experimental results show that our ranking model achieves high precision and recall.

Acknowledgments. This work is partially supported by the NSF (A#: 1219001).

6. REFERENCES

[1] Google Play Sound Search: *goo.gl/ahpvyO*.

[2] M. Bartsch and G. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *Trans. on Multimedia*, 7(1):96–104, Feb. 2005.

[3] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 12 1966.

[4] N. Bertin and A. D. Cheveigné. Scalable metadata and quick retrieval of audio signals. In *Proc. of the Int. Conf. on Music Info. Retrieval*, pages 238–244, Sept. 2005.

[5] R. Cai et al. Scalable music recommendation by search. In *Proc. of the Int. Conf. on Multimedia*, pages 1065–1074, Sept. 2007.

[6] P. Cano et al. A review of algorithms for audio fingerprinting. In *Workshop on Multimedia Signal Proc.*, pages 169–173, Dec. 2002.

[7] A. L. chun Wang. An industrial-strength audio search algorithm. In *Proc. of the Int. Conf. on Music Info. Retrieval*, Oct. 2003.

[8] M. Fink et al. Mass personalization: social and interactive applications using sound-track identification. *Multimedia Tools and App.*, 36(1-2):115–132, 2008.

[9] J. Foote. An overview of audio information retrieval. *Multimedia Syst.*, 7(1):2–10.

[10] R. Gray. Vector quantization. *Acoustics, Speech, and Signal Proc. Magazine*, 1(2):4–29, Apr. 1984.

[11] C. Herley. Accurate repeat finding and object skipping using fingerprints. In *Proc. of the Int. Conf. on Multimedia*, pages 656–665, Nov. 2005.

[12] K. S. Jones et al. A probabilistic model of information retrieval: Development and comparative experiments. *Info. Proc. and Management*, 36(6):779–808, Nov. 2000.

[13] Y. Ke et al. Computer vision for music identification. In *Proc. of the Conf. on Comp. Vision and Pattern Recog.*, pages 597–604, June 2005.

[14] W. Li et al. Robust audio identification for MP3 popular music. In *Proc. of the Int. Conf. on Research and Dev. in Info. Retrieval*, pages 627–634, July 2010.

[15] T. Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Info. Retrieval*, (3):225–331, 2009.

[16] D. Mitrovic, M. Zeppelzauer, and C. Breiteneder. Features for content-based audio retrieval. *Advances in Comp.: Improving the Web*, pages 71–150, Mar. 2010.

[17] A. Poritz. Hidden Markov models: a guided tour. In *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Proc.*, pages 7–13, Apr. 1988.

[18] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Info. Retrieval*, 3(4):333–389, Apr. 2009.

[19] D. Stowell and M. D. Plumbley. An open dataset for research on audio field recording archives: freefield1010. In *Proc. of the Int. Conf. on Semantic Audio*, Jan. 2014.

[20] R. Stratonovich. Conditional markov processes. *Theory of Probability & Its Applications*, 5(2):156–178, 1960.

[21] J. Tejedor et al. Comparison of methods for language-dependent and language-independent query-by-example spoken term detection. *Trans. on Inf. Syst.*, 30(3):18:1–18:34, Sept. 2012.

[22] A. Velivelli, C. Zhai, and T. Huang. Audio segment retrieval using a short duration example query. In *Proc. of the Int. Conf. on Multimedia and Expo*, volume 3, pages 1603–1606, June 2004.

[23] M. M. Zloof. Query by example. In *Proc. of the National Comp. Conf. and Expo.*, pages 431–438, May 1975.