Contemporary Music Review
Vol. 24, No. 6, December 2005, pp. 489–508

Routledge
Taylor & Francis Group

Acoustic Lexemes for Organizing Internet Audio

Michael A. Casey

In this article, a method is proposed for automatic fine-scale audio description that draws inspiration from ontological sound description methods such as Shaeffer's Objets Sonores *and Smalley's* Spectromorphology. *The goal is complete automation of audio description at the level of sound objects for indexing and retrieving sound segments within Internet audio documents. To automatically segment audio documents into acoustic lexemes, a hidden Markov model is employed. It is demonstrated that the symbol stream of cluster labels, generated by the Viterbi algorithm, constitutes a detailed description of audio as a sequence of spectral archetypes. The ASCII base-64 encoding scheme maps cluster indices to one-character symbols that are segmented into 8-gram sequences for indexing in a relational database. To illustrate the methods, the essential components of an audio search engine are described: the automatic cataloguer, the retrieval engine and the query language. The results of experiments that test the accuracy and the retrieval efficiency of six new similarity-matching algorithms for audio using acoustic lexemes are presented. The article concludes with examples of audio matching using the structured query language (SQL) for creating new musical sequences from large extant audio collections.*

Keywords: Acoustic Lexemes; MPEG-7; Audio Matching; Sound Object; Query Language

Introduction

General audio documents, including music, speech and environmental audio, are increasingly available on the Internet via large content repositories, the growth of which has been spurred by the development of derivative works licenses, such as the Creative Commons (Lessig, 2004). The scale of these resources necessitates the use of sophisticated tools for organizing and locating materials by the way their content sounds. Many approaches to content-based audio retrieval have been proposed (Wold et al., 1996; Zhang, 1998; Casey, 2002) that are based on searching in high-dimensional audio feature spaces and hence pose significant problems in realizing an efficient implementation (Berchtold et al., 2001). Here we propose a new method for

ISSN 0749-4467 (print)/ISSN 1477-2256 (online) © 2005 Taylor & Francis
DOI: 10.1080/07494460500296169

audio retrieval that represents high-dimensional audio features as strings describing the morphology of spectral archetypes.

Consider, for example, the open-source audio collection available at http://archive.org, which is a large collection of heterogeneous sound materials consisting of musical works, soundscape recordings, speeches, meetings and so forth, much of which is distributed with a Creative Commons deed. Among the audio resources are 23,690 live recordings made at concerts. The administrators estimate that there is about one Petabyte of data held in the archive, and while it is unknown what the total duration of audio is, we made an estimate using reasonable bounds. The lower bound on individual documents held in the archive is estimated as the total number of compressed video DVDs that would fit into the archive, about 10^5 or one hundred thousand full-length feature films. An upper bound for the duration of the audio archive is estimated as if the archive were entirely composed of MPEG Layer-III audio (mp3) files at 128,000 bits per second coding. With these limits, we estimate the total duration of audio on the archive to be somewhere between 34.2 and 2,000 years (Rhodes, 2005). Without content-based organization, the sounds within most of these documents will remain hidden. The current prevailing system of *artist*, *title*, *genre*, *date* textual term descriptions is not sufficiently detailed to enable access to sound events within the documents.

A second example of an Internet collection is the *Freesound* project (http://www.pfreesound.org). Here the focus of the collection is samples, individual sound objects or short source audio clips. The sound materials are organized into a database and can be retrieved using a non-specialized ontology based on the associative WordNet English lexical model (Miller, 1995). Within the *Freesound* project there is also content-based search capability that describes the 'microsound' structure using audio feature vectors influenced by the Shaefferian ontology. Each retrieved sound file can be used as a query to filter the database contents to the closest N matches, or the furthest N matches from the query. The current size of the *Freesound* database is 10,000 samples with a mean duration of 3.25 seconds per sample (Cano & Koppenberger, 2004).

Both of the aforementioned databases index whole audio documents: archive.org documents consist in large part of entire live recordings of concerts with multiple works occupying a single document, and, in the case of *Freesound*, the documents are generally short-duration samples intended to be used as a unit of musical creativity. These systems require that the audio is pre-segmented into individual documents that have a single level of description attached as required by the use case scenario. Likewise, most existing content-based audio systems, such as Wold et al. (1996) and Cano and Koppenberger (2004), provide a single audio feature vector for each document. Such systems are called 'homogeneous audio' search engines.

Given the temporal gulf between *concerts* and *sound objects*, a single level of description is not sufficient to access the content. For our purposes, we require retrieval of individual events located *within* audio documents. So, we aim to extract sound-objects from whole audio documents and make them individually available in a database as findable units for musical creativity. We require a method to automatically segment *heterogeneous* sound documents into homogeneous regions. Furthermore, we require database indexing methods so that these regions can be efficiently matched and retrieved in a query-by-example system. We call such systems 'heterogeneous audio' search engines (Casey, 2004).

We propose a system to automatically describe the fine structural detail of sound using symbolic representations of audio archetypes that enable content-based queries on sub-document level segments. Following Schaeffer's *Objets Sonores* and Smalley's *Spectromorphology*, our system uses ontologies of sound properties and sound behaviours (Schaeffer, 1966; Smalley, 1986). These description languages are applicable to diverse sonic materials whether they are pitched, un-pitched, noise-based, gesture, texture, single source or mixed. Like the aforementioned ontologies, *Acoustic Lexemes* describe sound spectra as sequences of archetypes; our description language differs in that the archetypes are extracted by machine learning from a set of sound examples. Acoustic Lexemes are derived from a corpus of audio material, such as a set of musical pieces or source materials, and compactly represent the salient acoustic behaviours within the corpus. One advantage of the lexeme representation is relative independence from theories about music thereby maintaining generality. In the following sections, we define acoustic lexemes and their interface with efficient relational database technologies that scale to searching in Internet volumes.

Creative Organization of General Audio

Content-based audio query systems admit a new type of music composition that we call 'meta-music', which is composed by querying large databases and synthesizing audio content from the results. For example, a variety of *audio mosaic* systems have been proposed that match segments from a target audio file to a database of audio source materials to synthesize a new version of the target from the source materials (Pachet et al., 2001; Casey, 2003; Schwartz, 2000; Sturm, 2004; Lazier & Cook, 2003). A composition emerges by concatenation of the retrieved audio output for each input segment. Control over the synthesis is achieved in a number of ways, such as changing the features, modifying the similarity function or by restricting the view on the source materials. The synthesis quality of these systems has been widely reported to improve as the size of the source database grows. However, as the volume of available source material grows, the computation time for retrieval grows exponentially. Therefore, efficient methods for automatic segmentation, indexing of long audio documents and retrieval of sub-segments have immediate application in such creative systems. We will now describe the essential components of our system. The first step was to decide which audio features to use.

Audio Features for Content-based Description

Choices of audio features and similarity measure crucially determine the attributes of self-similarity and therefore the definition of a segment. We sought features that could represent a wide variety of sound types; thus we excluded features that solely represented traditional music concepts such as pitch class and harmony. Instead, we used Cepstral Coefficients, which de-correlate timbre and pitch components of audio spectra. Taking the log of the spectrum amplitudes and applying the Discrete Cosine Transform (DCT) yields Cepstral features. We used a slightly modified form of the widely used Mel-Frequency Cepstral Coefficients (MFCC) employing a logarithmically spaced filter bank to approximate the psycho-acoustically motivated Mel frequency scale (Logan & Chu, 2000; Casey, 2004). Features are calculated from short windowed sub-sections of audio with each frame overlapping with the previous frame by some proportion of the window length. Overlapping windows by greater durations yields smoother features. The feature frame rate was 1/100th of a second, so at each 10ms time point an analysis window of 30ms duration was sampled and transformed to a feature vector as described above. The extracted features form a time-series trajectory that defines a probability space. The smoothness of features depends on the composition of sources, the window length, the hop size and the salience of the chosen features.

Modelling Spectral Dynamics

We adopted a data-driven approach using a 40-state hidden Markov model with parameters inferred by machine learning over a large corpus of audio training data. It is a reasonable assumption that certain sequences of spectra occur more frequently than others. We model such time dependence between audio features by a discrete first-order Markov chain process. In the model, the probability of the current active state, or *symbol*, depends solely upon the identity of the symbol in preceding step:

$$\mathrm{T}(j,k) \equiv \text{Probability(current\_symbol is } k \text{ GIVEN previous\_symbol was } j).$$

Therefore we condition the choice of current_symbol upon the previous_symbol using a discrete probability distribution over all possible sequences of two symbols, of which there are N^2 (see Figure 1).

Audio features are not usually available as discrete symbols, so how does the Markov transition model apply to the audio feature data? The answer is to organize the model hierarchically and treat the symbols as generators of audio features (Rabiner, 1989). We model these generators as random processes, by which we mean that a symbol k generates the observed audio feature at each time step by sampling from a multidimensional Gaussian density:

$$\begin{aligned}\mathrm{P}(\text{feature GIVEN current\_symbol} = k) &\equiv P\{\mathbf{x} \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}\\ &\qquad k \in \{A, B, \ldots, Z, a, b, \ldots, n\}\end{aligned}$$

where

$$N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = (2\pi)^{-\frac{d}{2}|\Sigma_k|^{-1/2}} \exp\left[-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)\right].$$

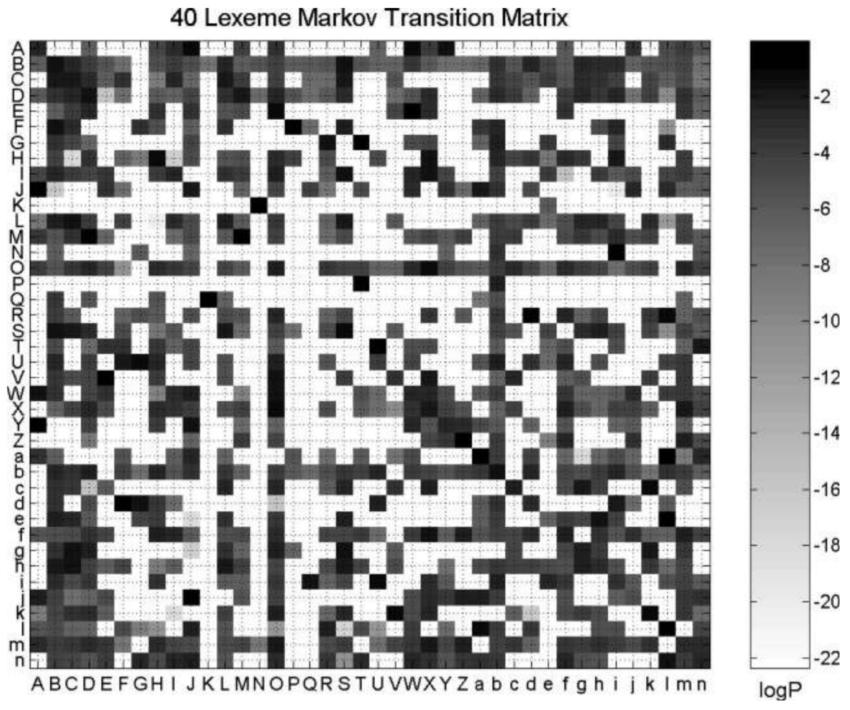

Figure 1 Transition matrix for a 40-state Markov model. Some transitions have effectively zero probability of occurrence and are therefore excluded from the set of two-symbol sequences generated by the model.

Putting the data-generating function and the symbol transition probabilities together we can write a hidden Markov model (HMM) for acoustic lexemes invoking Bayes rule for factoring our conditional probability distributions:

$$\mathrm{P}(\text{current} = k \text{ GIVEN feature AND previous} = j) \propto P(\mathbf{x} \sim N(\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))\mathrm{T}(j, k).$$

This equation leads to a method to assign the observed feature to a lexeme symbol by choosing the symbol k that maximizes the product of the two known distributions.

To train the HMM, we used a portion of the available data inferring the parameters to the set of generating Gaussian densities using maximum likelihood learning with the expectation maximization (EM) algorithm (Dempster et al., 1977). To test our proposed system, we trained a 40-state HMM using 200 randomly chosen 10-second clips chosen from 20 works in a varied corpus of electroacoustic music; there were 128 full length works in total, 20 used for training 108 for testing retrieval methods. This material was chosen for its diverse sound content and difficulty for segmentation tasks with materials consisting of vocal utterences, synthesized audio, environmental audio, concrete sources and instrumental sources structured with little pitch-time lattice organization. Figure 2 shows a view of the first two dimensions of a 20-dimensional Log Frequency Cepstral Coefficient (LFCC) feature space that is

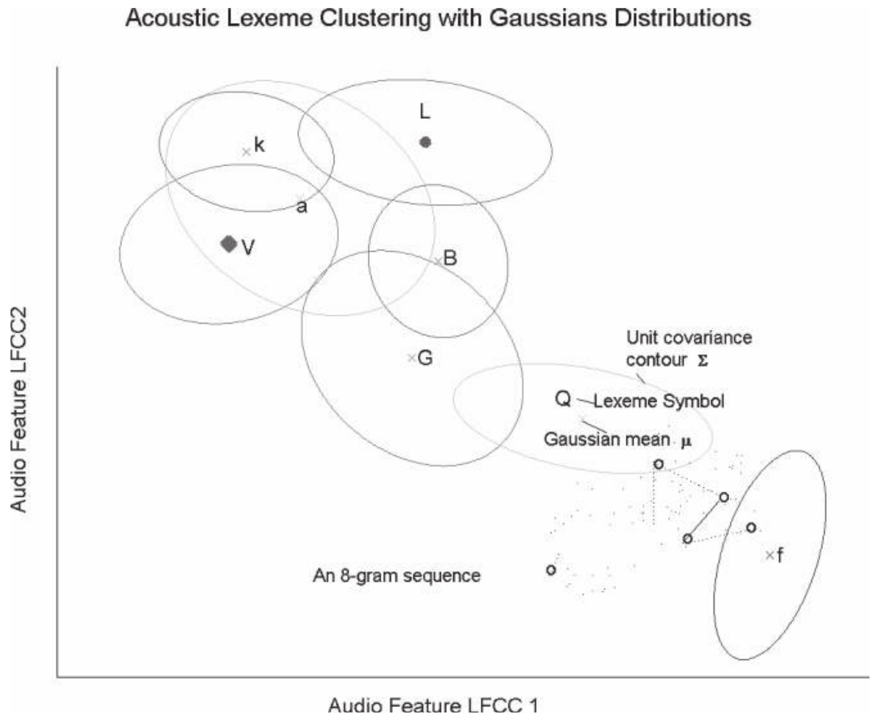

Figure 2 Clustering of the audio feature space by a mixture of Gaussian probability density functions. Here we show only 8 of 40 clusters for illustrative purposes. In a hidden Markov model, only one distribution is active per sample, so each audio feature vector gets assigned to one of 40 Gaussian lexeme symbols.

populated by Gaussian distributions over the data. The distributions describe the probability of each location in the feature space, given one of the states as the active class. The ellipses describe iso-contours of like probabilities.

Automatic Segmentation of Internet Audio

We implemented an automatic lexeme segmenter to label sequences of audio buffers, 10ms apart, using a pre-trained hidden Markov model and the Viterbi algorithm to estimate the most probable sequence of symbols that generate the given sequence of audio buffers (Viterbi, 1967) (see Figure 3). Repeated symbols are removed from the Viterbi symbol stream thus integrating 10ms input buffers into longer homogeneous regions. Repeats occur where consecutive audio buffers are labelled as samples drawn from the same multidimensional Gaussian random process (i.e., when there is no state transition). When a transition occurs, we label the segment with the acoustic lexeme corresponding to the associated Gaussian random processes, or HMM state. The symbols are represented using the ASCII base-64 encoding of the numerical index of each Gaussian state, 1-40. Figure 4 illustrates the mapping of Markov model states to ASCII.

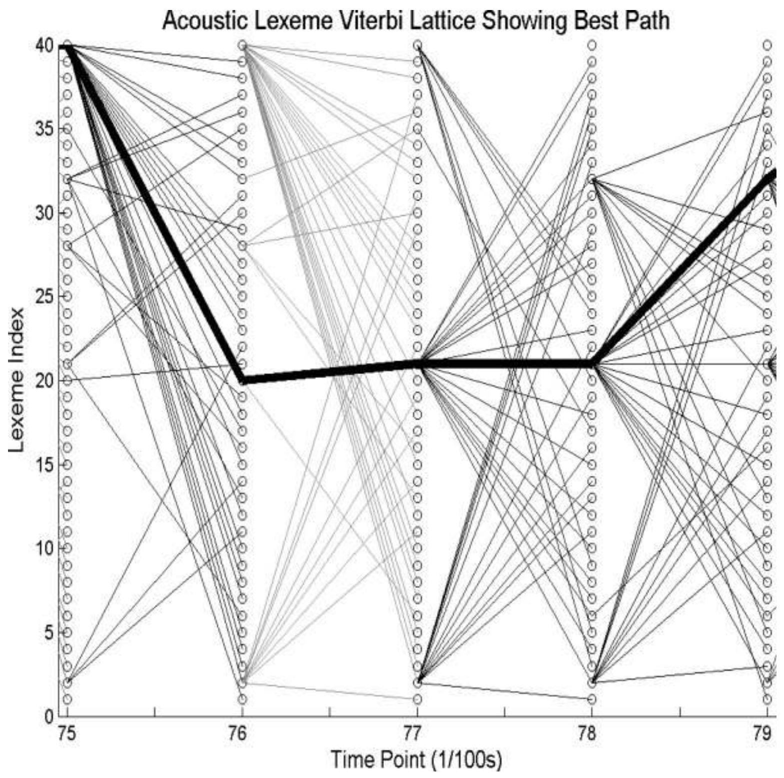

Figure 3 A Viterbi trellis used for solving the optimal state sequence from observed data in a hidden Markov model. At each time step, the colour of lines between states indicates the probability of transition as the product P(x|k)T(j,k) (see text). Lexeme symbol boundaries are determined by transitions between states. Lexical 'strings', consisting of two or more lexemes, are used to index sound segments in the relational database.

state	**1 2 3** **…** **40**
ASCII	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn

Figure 4 Mapping of state index to ASCII code symbol.

Figure 5 provides a system diagram of our *AutoSeg* system, which converts sequences of 20-dimensional LFCC audio feature vectors to lexeme N-gram sequences represented as strings of ASCII characters. Audio segments, then, are represented by strings in a relational database and are efficiently stored and retrieved using a B-Tree data structure that is built into most relational database management systems. As such, content-based filtering and sorting is extremely efficient because audio content search is transformed to string matching using well-known tree-based sequence matching algorithms such as those described by Sankoff and Kruskal (1983).

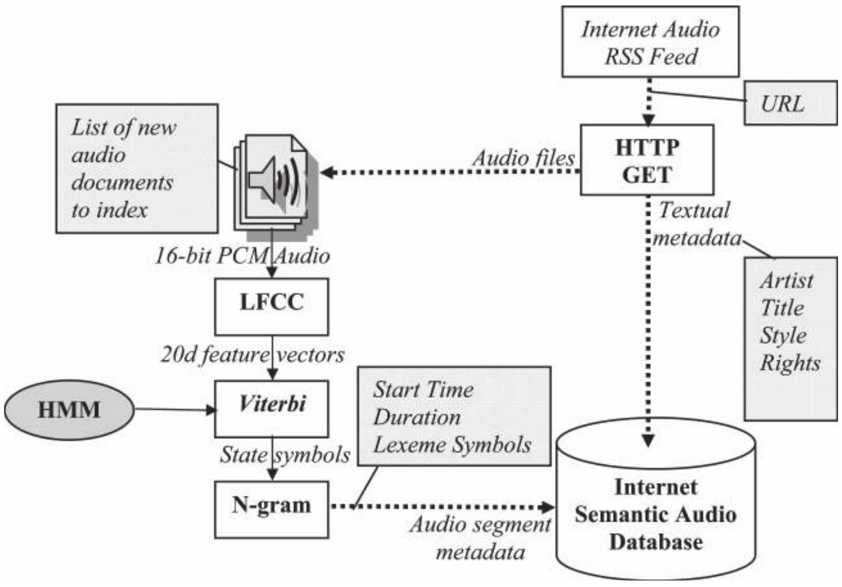

Figure 5 System for automatic segmentation and relational indexing of sub-document-level segment within audio files available on the Internet.

Lexeme N-gram Sequences

N-grams are ordered sequences of symbols that are fixed to a given length, N; chosen between 2 and 8 in our experiments. The AutoSeg segmenter allocates a new symbol only when a transition is generated by the Viterbi algorithm, thus repeated state symbols are joined into a single lexeme symbol with a duration attribute in were multiples of 10ms, the signal window rate. The segmenter accumulates duration into a total for each N-gram. After N symbols have been concatenated to form a string, they are inserted into the database along with their start time and N-gram duration. We extracted all 8-gram sequences from our 128-work database for our experiments. There were 1.6 million individual 8-gram segments for the 128 works. Lower-order N-grams, at orders 2 through 7, are sub-sequences of the 8-gram sequences and were accessed in our experiments via substring searches on the 8-gram sequences. In our database, the durations of the 8-grams varied from 0.08 seconds, with each symbol's duration equal to 1 feature vector (0.01s), to 52.05 seconds with each symbol representing many hundreds of feature vectors. We calculated that there were 742 unique bi-gram sequences and 669,340 unique 8-gram sequences out of 1.6 million sequences in the database of 128 electroacoustic works consisting of about 600 minutes of audio.

Interpretation of Automatically Extracted Lexeme N-Grams

Figure 6 illustrates the most frequently occurring 2-grams, or *bi-grams*, in our database. The first lexeme bi-gram represents a transition from wide band spectrum to a low-frequency spectrum; a characteristic of onsets. By listening, we verified that

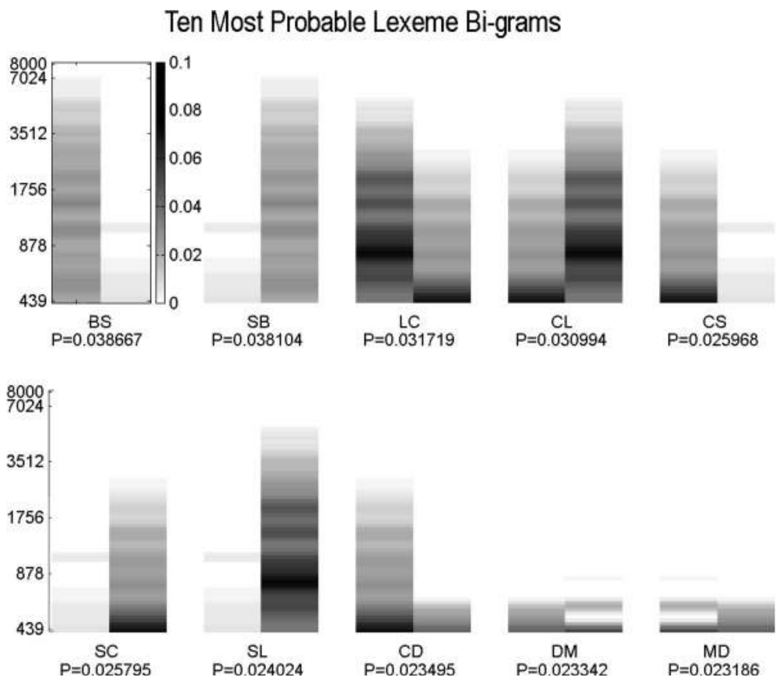

Figure 6 The ten most frequent lexeme bi-grams of 1.6 million shown as spectral prototypes; there is no time index, lexemes encode sequences of spectral archetypes, the order is encoded but not the duration. The ASCII Base64 Encoding of the bi-gram is given below each prototype. N-gram sequences, such as these bi-grams, are duration-independent. We encode duration separately in the database for maximum retrieval control.

these were indeed onsets. The distribution of energy in the archetype spectrum shows that these lexemes are selective for energy in the mid-range frequencies. On listening, we observed selectivity for mid-range pitched components around A440 with *reed*-like timbres. The second lexeme is the reverse of the first indicative of an onset occurring within a sustained segment.

The third lexeme bi-gram also consists of wide band to low-frequency transition, but these lexemes select for lower frequency components than the first two bi-grams. The fourth is the mirror image of the third and the fifth lexeme bi-gram shows low-frequency components decaying. The sixth bi-gram mirrors the fifth and also selects for low-frequency components. The next bi-gram represents transitions from low energy to high frequency with intensity. And the final two again are mirror images of low-frequency sustained tones with some undulation causing periodic state change.

There were a total of 742 such unique bi-grams occurring in the database of 1.6 million segments. The relatively low number of unique morphologies generated by the HMM suggests that there is a high degree of temporal structure in the audio data and supports our requirement of description compactness. The ten most common 8-gram sequences are shown in Figure 7. The first two 8-grams encode

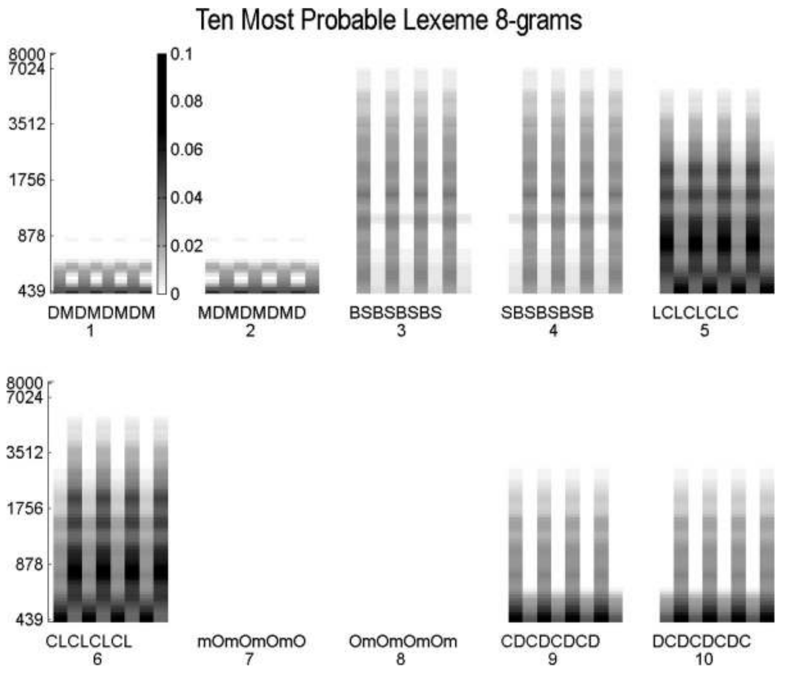

Figure 7 The first ten of 669,340 unique lexeme 8-gram sequences. As in the previous figure, the images show the sequences as prototype spectra using each lexeme's Gaussian mean approximation synthesized as a spectral frame. This illustrates the degree of repetition and patterning inherent within audio data. The most probable lexeme sequences occur many thousands of times in the database.

sustained low-frequency energy; they occur, respectively, 16,753 and 16,692 times with mean 8-gram duration of 0.2s. 8-grams 7 and 8 represent silence, with alternating low-energy spectral archetypes; they occur 5,228 and 5,217 times in the database, respectively, and have a mean duration of 0.25s.

These N-gram illustrations are just a few examples of the specific spectro-morphological *meaning* of symbolic lexeme sequences. In these cases we inspected, the most frequently occurring segments as classified by their bi-grams and 8-grams. Our observations reveal some interesting properties of the model. In particular, we observed a high degree of symmetry and structure in the resulting model in spite of the diversity of audio content. We now propose that these archetypal structures can be used to match sound segments simply by matching the strings that represent their archetype sequences. Our central hypothesis is that sound segments with similar strings will *sound* similar when presented to the human listener. A discussion and evaluation of the validity of this hypothesis is outside the scope of the current article (for detailed discussions on this hypothesis with respect to audio features and human similarity judgments, see, e.g., Pohle et al., 2005; Berenzweig et al., 2003).

We now go on to discuss how lexeme sequences are expressed in a relational model so that they can be accessed within a relational database for efficient retrieval.

Relational Data Model for Acoustic Lexemes

To implement our system, we used the relational database model thereby enabling use of proven industrial-strength, scalable database management systems such as *PostgreSQL* (Stonebraker et al., 1983). We implemented our system using standard distribution components with no specialized software for query processing and matching. Our implementation is centred upon two relations (tables) called the 'MediaTable' and the 'LexemeTable' (see Tables 1 and 2). The first table contains a unique locator for the media on the Internet using a uniform resource locator. It also stores any textual metadata that is attached to the whole document such as *artist*, *title*, *style*, *publication date*, and so on. The first table is linked to the LexemeTable via a unique identifier for the audio document. The LexemeTable contains the sub-segments that were automatically extracted for each document. Each segment is allocated a unique identifier, a start time, duration and an 8-gram sequence of acoustic lexemes stored as 8-character strings. We also represent histograms of the occurrence of characters within a 40-character string to support alternate matching methods described below.

Creative Queries with Acoustic Lexemes

As discussed above, a number of systems have previously been proposed that express creative musical processes as queries to a database (Zils & Pachet, 2001; Casey, 2003;

Table 1 Relational database *MediaTable* (audio documents)

Media relation	*Data type*	*Example entry*
mediaID	integer key	3821* (primary key)
URL	text	"http://sounds.org/file01.wav"
Title	text	"Source recordings"
Author	text	"M. Casey"
Date	text	"30 May 2005"
Format	text	"WAVE RIFF"
Description	text	"ambient cityscape recorded in barcelona"

Table 2 Relational database *LexemeTable* (audio segments)

Lexeme relation	*Data type*	*Example entry*
segID	integer key	87492* (primary key)
mediaID	integer key	3821* (foreign key)
lexemes	character(8)	"LTLjajLT"
startTime	integer	41548 (centi-seconds)
stopTime	integer	42120 (centi-seconds)
duration	integer	572 (centi-seconds)
histogram	character(40)	"0000000000030000000200000010000000020000"
indicator	character(40)	"0000000000010000000100000010000000010000"

Schwartz, 2000; Sturm, 2004; Lazier & Cook, 2003). All of these systems start with some extant material to be used for querying. This audio is matched, segment-by-segment, against a database of pre-indexed sound material. In such systems, retrieved segments may be in any specified relationship to the target audio, such as nearness or farness in feature spaces. The nature of such queries can get very complicated, so to express organization of audio in a rigorous and robust manner we must use a query language.

Many composing languages have been proposed, mostly based on the orchestra/score synthesis paradigm, such as *Music-IV* and *CSound*, or the event-based systems such as *PureData* and *Max-MSP*. In none of these systems can a query such as those posed above be expressed succinctly as a concept. Fortunately, specialized query languages exist that are used to express database queries and optimize the query based on algebraic re-writing of it. In the future, it is unlikely that composers will learn complex query languages. Instead, the query operations will be embedded within software and a simple-to-use graphical user interface will be used to construct a query.

One such language is the Structured Query Language (SQL) in which a number of basic operations, such as INSERT and SELECT, are combined to form more complex and expressively powerful queries. SQL assumes the relational model for the data, so we use the relations defined above that represent both tables and columns for each media item and N-gram segment (Stonebraker et al., 1983).

Figure 8 illustrates the query framework for segment retrieval from Internet audio. Feature extraction is applied to the query audio and the features are then passed through to the Viterbi algorithm. The Viterbi state sequence is then post-segmented by removing repetitions and recording each state index and its duration. The symbol stream is ASCII base-64 encoded and concatenated into 8-gram sequences which are wrapped in an SQL command and passed to the query execution program (see Figure 9). Here, lexeme matching is applied to a pre-stored database of acoustic lexeme segment descriptions *from the same lexicon*. The best matches are returned as a list of URLs, start times and durations that are then fetched via the Internet and presented to the user within their application. As well as lexeme processing, the query engine can integrate high-level textual metadata within the SQL query framework, thereby enabling constraints on which portions of the database are used for matching.

Structured Query Language for Audio Retrieval

Systems for *audio mosaicing* perform matching using search over audio feature vectors. Such searches consist either of pre-computing *N*N* distances, which is intractable for very large N, or employing multidimensional indexing structures such as R-Trees or KD-Trees. Unfortunately, there are no standard methods to support vector-based queries in dimensions higher than two, and there is little or no support for vector-based queries in available database management systems. Furthermore, high-dimensional vector space indexing does not inherently represent sequences; therefore, further research is required before efficient search methods for

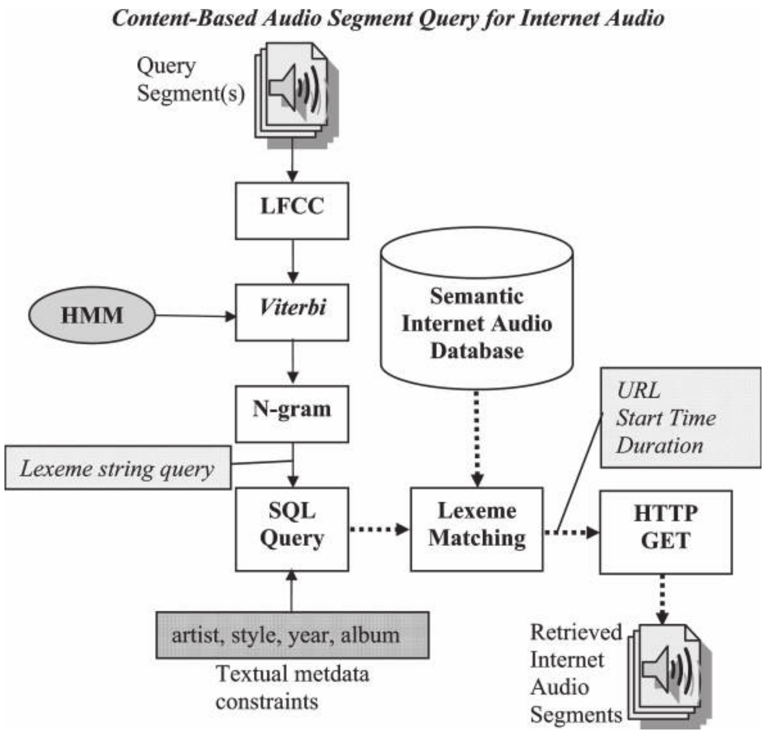
Content-Based Audio Segment Query for Internet Audio
Query Segment(s)
LFCC
HMM
Viterbi
N-gram
Lexeme string query
SQL Query
Semantic Internet Audio Database
Lexeme Matching
URL
Start Time
Duration
HTTP GET
artist, style, year, album
Textual metdata constraints
Retrieved Internet Audio Segments

Figure 8 Schematic diagram of the lexeme retrieval system. Audio queries are presented via a graphical user interface, such as a sound editor or media player, low-level audio features are then extracted and passed to the HMM Viterbi decoder. The string of symbols is converted to an 8-gram and used to query the semantic audio database. The result is a set of media locators for audio segments that match the query criteria.

high-dimensional vector sequences become widely available. Until these problems have been solved, we propose a range of query templates that can be used with most available relational database management systems utilizing the acoustic lexeme representation to perform approximate spectral matching in a large-scale audio database. We now give some examples of how segment matching is expressed in the structured query language (SQL) using the relations defined above.

Duration Only Constraint

In the first example, we employ the SELECT command to retrieve all segments in the database that are less than 10 and greater than 2.5 seconds in duration:

```
SELECT segID,mediaID FROM LexemeTable WHERE duration < 1000 AND
duration > 250;
```

This command returned 1,620 rows out of 1.6 million in our database, thus filtering the database to about 1/1000th of the original entries.

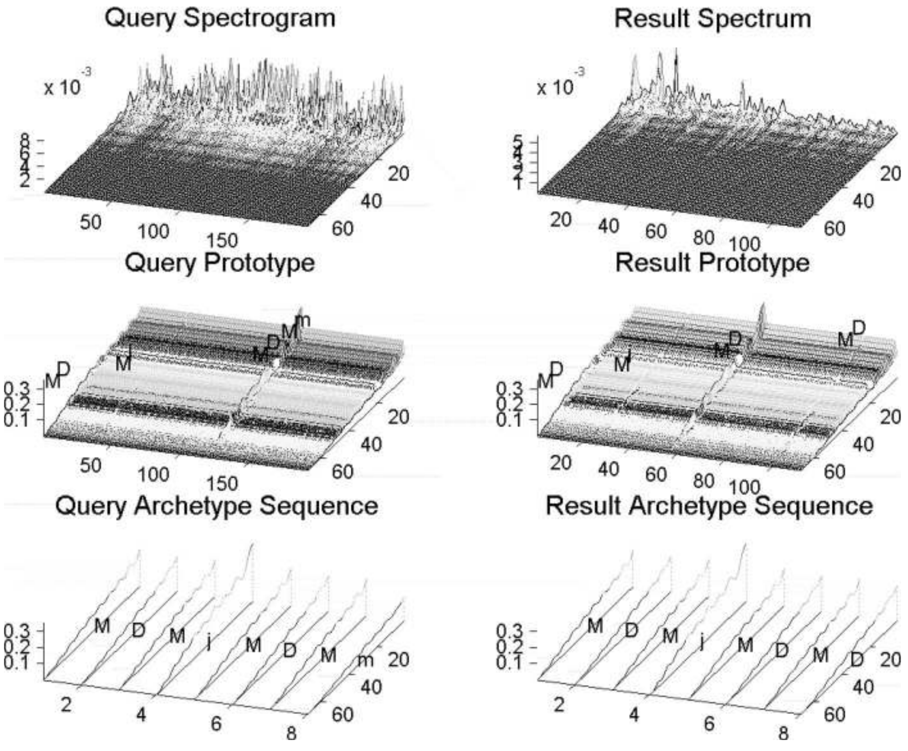

Figure 9 Three layers of the query and retrieval process using acoustic lexemes. The top layer shows spectrograms of the query and result with time in 1/100th second and 1/16th-octave logarithmic frequency bands indexed from 1 to 70. The second layer shows the time-aligned prototypes synthesized by the hidden Markov model for each spectrogram. The bottom layer shows the compact lexeme 8-gram sequence for each of the query and result prototypes. Matching is performed with the bottom layer, which is agnostic to time.

The table columns for the LexemeTable include duration, segID and mediaID and they are defined, along with the other columns, in Table 2 above. It is noteworthy that the duration here refers to the total duration of the lexeme sequence (8-gram) and not the duration of individual lexemes as explained above. In the following example, we apply further constraints to filter out more of the database to retrieve more targeted results.

Lexeme Exact Match

We first write an SQL expression to filter the database to those segments that have the exact same sequence of eight lexemes as the query segment:

```
SELECT segID,mediaID FROM LexemeTable WHERE lexemes= 'TjTjZjZj'
AND duration < 1000 AND duration > 500;
```

The chances of an exact match are defined by the probability distribution that lexeme sequences are drawn from. It is not guaranteed that there will be an exact match in the database to a given lexeme sequence. If there is no exact match, then we must invoke one of a number of approximate matching methods. The most probable lexeme sequence, DMDMDMDM, is associated with 16,753 segments in different audio files in our database of which 21 are between 2.5 and 10 seconds in duration. Thus the length of our result list is reduced to 21 candidates matching our search criteria from a total of 1.6 million initial candidates.

Lexeme Substring Match (Low-order N-gram Matching)

The following query truncates the query string and performs matching only against the shortened lexeme string. If the shortened result string matches the shortened query string, then the segment is returned as a match.

```
SELECT segment FROM table WHERE lexemes LIKE 'TjTj%' AND duration <
1000 AND duration > 500;
```

Lexeme Sequence Approximate Matching

Given the probabilistic nature of our model it is likely that similar *sounding* segments have similar, but not identical, lexeme sequences. Using the histogram indicator function of our lexeme 8-gram sequences we can make an exact string-match query that generates an approximate match by reducing constraints on symbol ordering when matching.

Lexeme Histogram Match

The first type of approximate matching that we considered was to match using histograms of eight-lexeme sequences. The maximum number of occurrences of a lexeme is four since our representation has eliminated repetitions. Our encoding placed a digit 0 – 4 in each lexeme position [A,B, . . . , Z,a,b, . . . , m] therefore making a 40-character string: one character for each possible lexeme (see Table 2).

```
SELECT segID,mediaID FROM LexemeTable WHERE histogram=query-
Histogram AND duration < 1000 AND duration > 500;
```

The matching algorithm then considered only those segments with exactly matching lexeme histograms. Clearly this includes the set of segments returned by an exact match, since exact sequence matches will have matching histograms too. However, the filter generally admits many more sequences than the extract lexeme sequence string match.

A second approximation was to use the indicator function of the histogram. This lexeme key is almost the same as the histogram, except that a 1 is used for every non-zero histogram value, indicating that the state is active during the segment.

This lexeme symbol indicator function returns all segments in the histogram match, but it also admits many more sequences because the precise counts are not matched. Motivated by the overlap between the result lists of the histogram matching methods, in our experiments we used the histogram indicator matching method rather than the more restrictive matching symbol count method.

Lexeme String Distance

A third and final approximate matching scheme was to compute a distance between the query string and lexeme strings stored in the database. The canonical distance metric that is used for strings is the Levenshtein distance, or string-edit distance (Sankoff & Kruskal, 1983). This quantity can be computed in a number of ways: we used a dynamic programming method to implement the Levenshtein distance.

```
SELECT segID,mediaID FROM LexemeTable WHERE

stringDist(lexemes,'TjTjZjZj') < 2

ORDER BY abs(duration-queryDuration);
```

Results

The retrieval experiments reported below are for randomly chosen sub-segment queries from the set of 108 complete electroacoustic works that formed our testing set. We used indexing on the columns to be matched. Indexing is a technique to explicitly represent the relationships between data in relational tables by pre-sorting the data entries into an efficient data structure. We used a B-Tree index on each column that appears in the queries described above. Use of a one-dimensional key representation for indexing audio is analogous to an Internet search engine that uses pre-computed indexes over the content of all webpages to efficiently locate documents containing a given text.

Figure 10 shows the results of performing 100 queries with randomly chosen segments from the database. The query excluded the source documents from which queries were drawn. The evaluation of the result is given in terms of the mean distortion between the query and the 1-best result drawn from the database of 1.6 million segments. Distortion was computed using a probabilistic measure of the divergence between query and retrieved spectrograms. We can see there is a significant difference between the six proposed algorithms.

The most significant difference is between the anchored substring algorithms and the rest. Anchored substrings were chosen as 4-grams. The task was to retrieve a segment that matched the query 8-gram using only the first 4 symbols for the match. Thus the divergence is high because there are no constraints on the last four symbols. The difference between the Levenshtein distance approximate match methods and

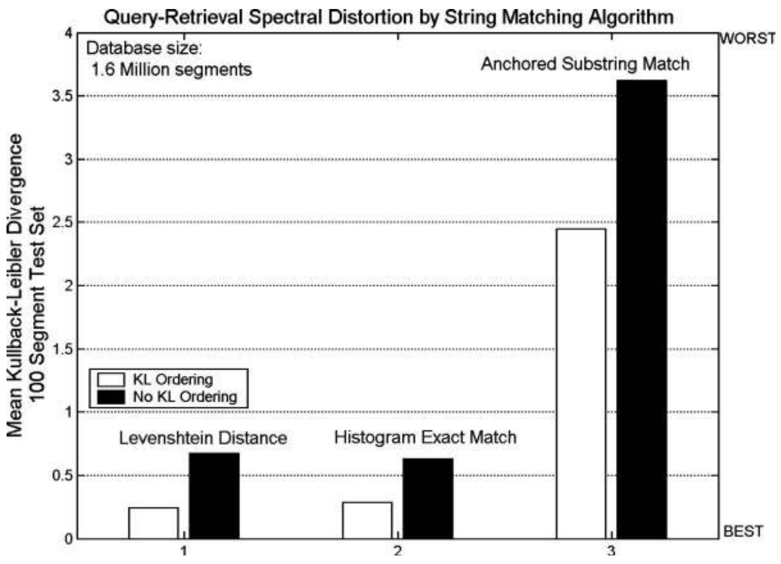

Figure 10 Evaluation of string-matching algorithms by spectral distortion between 100 query segments and corresponding retrieved. The Levenshtein distance was applied to a list of duration constrained row in order to minimize distance computations. These results show that histogram matching performs as well as approximate string matching using the Levenshtein distance; substring matching does not perform well.

the histogram exact match method are insignificant. Thus we have shown that lexeme approximate matching can be computed efficiently using un-ordered, un-counted histogram representations that yield similar spectral divergence properties to approximate string matches but are far more efficient. The temporal efficiency of the search methods is illustrated in Figure 11. Matching by Levenshtein distance was the most time-consuming method even though the database was pre-filtered in this case to include only those segments with durations within a factor of two of the query segment. The most time-efficient method was the histogram indicator exact match method, which, as stated above, also yielded good spectral distortion performance.

The retrieval times for our queries using the lexeme histogram approximate matching methods benefited from the B-Tree indexing scheme, which is known to have a theoretical retrieval performance of order O(log(N)) in the number of segments N. This means that retrieval times will grow by, approximately, the log of the number of segments in the database. For example, in our experiment we recorded retrieval times of 0.01s for 600 minutes of audio. We can extrapolate this result to the estimated size of Internet audio, 10^10 minutes of audio; all else being equal, the retrieval times would scale to approximately 0.5 seconds with a conservative allowance for estimation errors in our calculations. This result demonstrates that our methods are scalable with

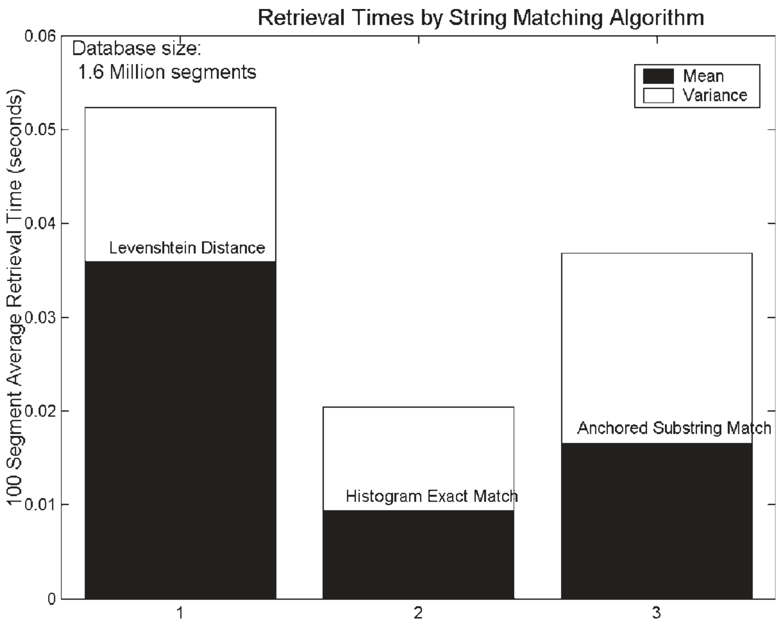

Figure 11 Evaluation of string-matching algorithms by mean retrieval times for 100 queries to the audio database. These results show that the histogram exact match method was the most time-efficient algorithm.

reasonable time complexity to sound-object searches within extremely large audio databases such as those that archive all Internet content.

Conclusion

In this article we have proposed a framework for articulating the internal materials of audio documents, regardless of their content and duration, in relational database management systems via the use of a new mid-level audio representation called 'acoustic lexemes'. We discussed the importance of sub-document indexing at the sound-object level and proposed a machine-learning framework to automatically derive a vocabulary of audio archetypes from a corpus. We demonstrated how the archetypes can be used to automatically segment audio into spectromorphological sequences represented by N-grams of acoustic lexeme symbols.

We proposed a number of query strategies that use the lexeme N-gram representation in a relational data model using the structured query language and evaluated these strategies by the distortion of the retrieved segments and the time efficiency of the retrieval. We demonstrated that the best performing method in our framework also scales well in the size of the database having logarithmic search complexity in the number of segments in the database.

In the next few years we are likely to witness new modes of creativity that extend concepts such as the meta music models described herein, drawing on vast extant resources using sophisticated search and retrieval methods. Such systems draw from decades of knowledge from diverse intellectual disciplines. It is hoped that the bringing together of such knowledge sources into a unified computational creative framework will encourage future exploration by both artists and scientists, and will act as a catalyst for collaboration. Our future research plans include further development of automatic segmentation methods, exploring the emergence of sound objects and gestures in different corpora, and the development of intelligent performance systems that respond to input in real time by drawing on vast extant musical knowledge represented in an efficient database such as the one reported herein.

Acknowledgements

The author would like to thank Dr Christophe Rhodes at Goldsmiths College, University of London, for his insightful comments and, in particular, for his calculation of the approximate total duration of Internet audio. This research was supported in part by EPSRC Grant number GR/S84750/01.

References

Berchtold, S., Bohm, C. & Keim, D. (2001). Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *Journal of the ACM*, *33*, 322 – 373.

Berenzweig, A., Logan, B., Ellis, D. P. W. & Whitman, B. (2003). A large-scale evaluation of acoustic and subjective music similarity measures. Paper presented at the 4th International Conference on Music Information Retrieval, Baltimore, MD.

Cano, P. & Koppenberger, M. (2004). Automatic sound annotation. Paper presented at the 14th IEEE Workshop on Machine Learning for Signal Processing, São Luís, Brazil.

Casey, M. (2002). General sound classification and similarity in MPEG-7. *Organized Sound*, *6*(2), 153 – 164.

Casey, M. (2003). Sound replacement, beat unmixing and audio mosacing: Creative uses of MPEG-7 audio. Tutorial session at the Conference on Digital Audio Effects (DAFX03), London.

Casey, M. (2004). General audio information retrieval. In R. Raieli & P. Innocenti (Eds), *Multimedia information retrieval.* Rome: AIDA.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* Ser. B., *39*, 1 – 38.

Lazier, A. & Cook, P. (2003). Mosievius: Feature-driven interactive audio mosaicing. Paper presented at the 6th International Conference on Digital Audio Effects (DAFx-03), London.

Lessig, L. (2004). *Free culture: How big media uses technology and the law to lock down culture and control creativity.* New York: Penguin Press.

Logan, B. & Chu, S. (2000). Music summarization using key phrases. Paper presented at the International Conference on Acoustics, Speech and Signal Processing, Istanbul.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39 – 41.

Pachet, F., Roy, P. & Cazaly, D. (1999). A combinatorial approach to content-based music selection. Paper presented at the IEEE International Conference on Multimedia Computing and Systems, Florence, Italy.

Pohle, T., Pampalk, E. & Widmer, G. (2005). Evaluation of frequently used audio features for classification of music into perceptual categories. Paper presented at the 4th International Workshop on Content-based Multimedia Indexing (CBMI'05), Riga, Latvia.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257 – 286.

Rhodes, C. (2005). Personal communication, Goldsmiths College, University of London.

Sankoff, D. & Kruskal, J. (1983). *Time warps, string edits and macromolecules.* Boston, MA: Addison-Wesley.

Schaeffer, P. (1966). *Traité des Objets Musicaux.* Paris: Editions du Seuil.

Schwarz, D. (2000). A system for data-driven concatenative sound synthesis. Paper presented at the Conference on Digital Audio Effects (DAFX00), Verona, Italy.

Smalley, D. (1986). Spectromorphology and the structuring process. In S. Emmerson (Ed.), *The language of electroacoustic music.* Basingstoke: Macmillan.

Stonebraker, M. et al. (1983). Document processing in a relational database system. *ACM Transactions on Information Systems*, *1*(2), 143 – 158.

Sturm, B. L. (2004). MATConcat: An application for exploring concatenative sound synthesis using MATLAB. Paper presented at the International Computer Music Conference, Miami, FL.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *IT-13*, 260 – 269.

Wold, E., Blum, T. & Keislar, D. & Wheaton, J. (1996). Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, *3*(3), 27 – 36.

Zhang, T. & Kuo, C. (1998). *Content-based classification and retrieval of audio.* Paper presented at the 43rd SPIE Annual Meeting on Advanced Signal Processing Algorithms, Architectures and Implementations VIII, San Diego, CA.

Zils, A. & Pachet, F. (2001). Musical mosaicing. Paper presented at the Conference on Digital Audio Effects (DAFX01), Limerick, Ireland.