A Comparison of Language Modeling and Probabilistic Text Information Retrieval Approaches to Monophonic Music Retrieval

Jeremy Pickens
Department of Computer Science
University of Massachusetts
Amherst, MA 01002 USA
jeremy@cs.umass.edu

Abstract

With interest in music information retrieval increasing the need for retrieval systems unique to music is also growing. Despite its unique properties music shares many similarities with text. The goal of this paper is to explore some of the capabilities and limitations of current text information retrieval systems as applied to the task of music retrieval. Monophonic music is converted into text and retrieval experiments are run using two different text information retrieval systems in various configurations. Finally, we will discuss whether the techniques applied here are generalizable to the larger problem of polyphonic music retrieval.

1 Introduction

The modern field of information retrieval (IR) began in the 1950s with the aim of using computers to automatically search collections of unstructured online text. Since that time, and particularly in the last decade with the popular arrival of the Internet, interest in varied, "multimedia" information retrieval applications has exploded. Not only has text retrieval branched out by incorporating audio speech recognition techniques, but image retrieval and video retrieval have received considerable attention. Music information retrieval is a third.

Text IR has a large head start on these other retrieval systems. Techniques have been intensively studied and refined for about four decades. Methods and models for music IR, on the other hand, are still in their infancy (Byrd & Crawford, 2000). Music retrieval will continue to mature. However, rather than let it mature in isolation, the attitude taken in this paper is that we should leverage traditional information retrieval work and examine the extent to which techniques developed for text can be applied to music.

Two primary reasons we can take this approach have to do with the assumptions we make about our music representation format. First, instead of digitized audio, we assume explicit knowledge of the musical structure of a piece of music. In other words, the various notes and their pitches, start times, and durations are known. (General audio music recognition, somewhat akin to speech recognition for text, is an unsolved problem and will not be addressed here.) This structure, or "music text', could be anything from time-stamped MIDI to full music notation, such as the Nightingale music notation format (AMNS, 2000). Second, we assume that the music we are searching is monophonic. Only one pitch is "active" at any given time slice. With these two assumptions, it will be possible to apply text retrieval techniques to music.

2 Background and Related Work

2.1 Inference Networks (Inquery)

The first text retrieval system under consideration is Inquery (Callan, Croft, & Harding, 1992; Turtle & Croft, 1991). This is a probabilistic retrieval system, based on the generalized framework of a Bayesian inference network. A Bayesian net is a directed acyclic graph. The nodes in the graph represent propositional variables and the arcs represent dependencies between those variables.

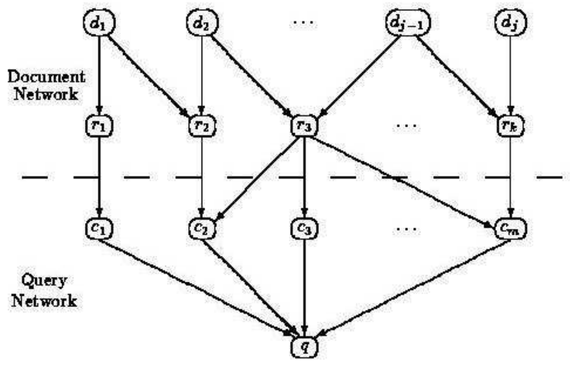

Figure 1: An example inference network for document retrieval

As Callan *et al.*, 1992 explain, the Inquery Bayesian net is divided into two principal components: the document network and the query network. The document network has two layers. The first layer is composed of document nodes (d_i). The document nodes are the leaves, and are the only observables in the entire system. There is one node for each document in a collection and the value of the node is the proposition that this document satisfies a user's information need. This value is always set to true for every document. The internal nodes in the graph, on the other hand, are not directly observable and must be inferred from the documents.

The second layer in the document network is composed of document content representation nodes (r_k), which represent the proposition that a given concept has been observed. As a simple example, one representation of one part of a document might be a word, a space- or punctuation-delimited sequence of alphanumerics. Other possible representations include integer values, dates, named entities and so on. One document can have multiple representations, and one representation can be shared by multiple documents. For example, a string of numbers (such as "1225") could be treated either as a text string or as an integer value or both, in which case two representation nodes would be used. Keep in mind that the concept in a representation node is not directly observable; it must be inferred from the document. The arc between a document node (d_i) and a representation node (r_k) is the conditional probability $P(r_k|d_i)$. The prior probability of any document $P(d_i)$ is $\frac{1}{N}$, where N is the number of documents in the collection.

The document content representation nodes connect with the first layer of the query network at the query concept nodes. The query network is a representation of a user information need. A representation node r_k can take on two values, true or false. The arc between the r_k and a query concept node c_l is the belief in the proposition r_k, the proposition that the concept r_k has been observed in the document. Bayes' rule is used to infer beliefs about document representation nodes r from documents, then beliefs about query concept nodes c from document represenation nodes. Just as multiple arcs are allowed from document nodes to document content representation nodes, multiple arcs are also allowed between document content representation nodes and query concept nodes. This will be useful later in the construction of phrases, in which multiple words are combined in some desired manner to create a single *phrase* query concept node.

The root of the tree, the node to which all arcs eventually lead, is the query node itself, q. This represents the proposition that the user's information need is met, and is always set to true. The various concept nodes c which comprise a query are combined using a weighted summation operator, summing the beliefs at each of the query concept nodes. Documents are ranked by this sum. The higher the belief that a given document meets the information need, the higher it is ranked.

2.2 Language Modeling

A second, less well-known probabilistic approach to text information retrieval is language modeling. The primary difference in this approach is that, unlike most other models in which the document indexing and document retrieval tasks are dissimilar, language modeling combines indexing and retrieval into a single model. The language model is used not only as an index, but as a method of estimating the probability of generating a query. Unlike the Inquery inference network approach, concepts are not inferred from documents as an intermediate step. The language modeling approach deals with the probabilities of generating the query words themselves, directly from the documents.

The language model we will use for music retrieval is the system developed by Ponte & Croft, 1998. For a given user query, documents in the collection are ranked by their probability of generating that query. This ranking formula, defined for each document model M_d in the collection, is a product of the probabilities of generating each term in the query, times the product of the probabilities of not generating all the terms that are not in the query:

$$\hat{p}(Q|M_d) = \prod_{t \in Q} \hat{p}(t|M_d) * \prod_{t \notin Q} 1.0 - \hat{p}(t|M_d) \tag{1}$$

The probability of generating a term, given the model of the document, is given by:

$$\hat{p}(t|M_d) = \begin{array}{ll} \hat{p}_{ml}(t|M_d)^{1.0-\hat{R}_{t,d}} * \hat{p}_{avg}(t)^{\hat{R}_{t,d}} & \text{if } tf_{(t,d)} > 0 \\ \frac{cf_t}{cs} & \text{otherwise} \end{array} \tag{2}$$

When the term is found within the document ($tf > 0$), we can use the first probability estimate. There are two part to this estimate: $\hat{p}_{ml}(t|M_d)$ and $\hat{p}_{avg}(t)$. The first part, $\hat{p}_{ml}$, is the maximum likelihood estimate of the term given the distribution of that term in the current document, M_d. This is simply the frequency of occurrence of the term in that document normalized by the length of the document. The second part, $\hat{p}_{avg}$, is the average over all documents *in which t occurs* of the maximum likelihood estimate of that term in each document. Finally, $\hat{R}$ is a risk function. Details are found in the Ponte & Croft, 1998 paper, but it is enough to say that $\hat{R}$ estimates to what degree the frequency of the current term t in a document varies from the normalized mean across all documents. The more this frequency differs from the mean, the riskier it is to use the maximum likelihood estimate, and the safer it is to use the average estimate. Hence, $\hat{R}$ is a mixing parameter between $\hat{p}_{ml}(t|M_d)$ and $\hat{p}_{avg}(t)$.

When the term is not found within the document, the best we can do is to use an estimate based upon all documents in the collection. Thus $\frac{cf_t}{cs}$ is simply the number of times that term appears in the collection as a whole, over the size of (number of documents in) the collection.

Our ranking formula Although the ranking formula used by Ponte & Croft, 1998 includes an estimate for all terms which do not appear in the query, our experiment does not consider these terms important. We are looking for specific melodies in our searches, and the notes that we are not looking for have little bearing on whether or not a melody is present in a document. Preliminary results (not shown here) indicate much better performance when the non-query terms are ignored, but further analyses need to be done before we are confident of this. Nevertheless, this is the intuition we have developed and so we use the following simplified language model ranking formula:

$$\hat{p}(Q|M_d) = \prod_{t \in Q} \hat{p}(t|M_d) \tag{3}$$

3 Experimental Design

3.1 Meldex Collection

The music document collection we used consists of almost 9,400 North American, British, Irish and German folksongs, German ballads, and Chinese ethnic and provincial songs (Schaffrath, 1995; DT, 1996; McNab, Smith, Bainbridge, & Witten, 1997). These folksongs are all monophonic; notes are of various durations, but not more than a single note is sounded at any given time slice. Ignoring durations, McNab *et al.*, 1997 show that seven contiguous note pitches (or six contiguous pitch intervals) are necessary, on average, to uniquely identify a song.

We will not be working with contiguous notes of this length (as explained in Sections 3.2 and 4.1), but it is important to note this limit and how it might affect our results. If we created sequences that were too long, our evaluation might be useless; the very length of the sequences would be enough to achieve good retrieval results. Interested readers can find a detailed analysis of the infometric properties of this collection in Downie, 1999.

3.2 Document Creation

Now that we have a collection we must convert our collection documents and queries into a form suitable for evaluation by our two text retrieval systems.

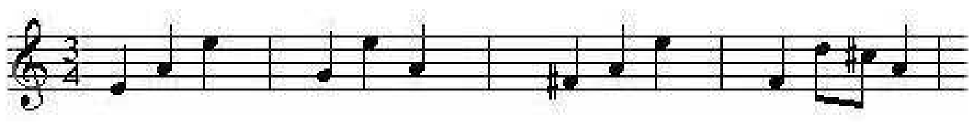

Figure 2: Lucy in the Sky with Diamonds

As an example we have the piece of music given by Figure 2 (Lennon & McCartney, 1967). We start with an abstract representation of this piece which includes the following information: time signatures, key signatures, note pitches and note durations. From this representation we construct the graphic in Figure 2. From this representation we also contruct a text document which represents this same piece. We also use this abstract representation to construct text queries with which we will test the effectiveness of our retrieval systems.

In order to convert music documents to text documents, some simplifying assumptions need to be made. The main assumption is that, while important, note durations are not nearly as important as note pitches. We therefore ignore duration as well as rests when converting music to text. Furthermore, we make the same assumption as other researchers that pitch intervals are better features than absolute pitches, because of possible transpositions from one key to another (Downie, 1999).

Unigrams The first type of pitch interval we use is interval unigrams. For example, the first note in Figure 2 is an E. The second note rises to an A. Subsequent notes rise to an E, fall to a G, rise to an E, then fall to an A. Pitch intervals are measured in semitones. So the sequence of pitch intervals begins:

$$\{+5, +7, -9, +9, -7, \ldots\} \tag{4}$$

In the Meldex collection there are no intervals larger than +24 or smaller than -24. For simplicity and to avoid working with "negative" text terms, we add 25 to each interval. Thus, the pitch interval unigrams range from 1 to 49. The song in Figure 2 becomes the following text document:

$$< doc1 > 30\ \ 32\ \ 16\ \ 34\ \ 18\ \ \ldots < /doc1 > \tag{5}$$

Bigrams The second type of pitch interval used is interval bigrams. Whereas unigrams only considered two contiguous notes, bigrams turn three contiguous notes into a single "text" term. The song in Figure 2 begins with the following ordered pairs of unigrams:

$$\{(+5, +7), (+7, -9), (-9, +9), (+9, -7), (-7, \ldots), \ldots\} \tag{6}$$

Using the same convention for eliminating "negative" terms as we did with unigrams, this becomes:

$$\{(30, 32), (32, 16), (16, 34), (34, 18), (18, \ldots), \ldots\} \tag{7}$$

We can again take advantage of the fact that the vocabulary size of the collection is limited and known. Consider an arbitrary ordered unigram pair (x, y). Neither x nor y are going to be larger than 49 or smaller than 1. Thus, we can use the following formula to convert (x, y) into a unique single value bigram:

$$bigram = 49x + y \tag{8}$$

This finally yields the following bigram "text" document:

$$< doc1 > 1502\ \ 1584\ \ 818\ \ 1684\ \ \ldots < /doc1 > \tag{9}$$

Note that even though these converted text terms are numbers, it is important not to think of them as integers, in the sense that they can be ordered, added, subtracted and so on. The term "1226" = *(25, 1)* might actually be more similar to "1177" = *(24, 1)* than it is to "1225" = *(24, 49)*, even though the integer 1226 is closer to the integer 1225. The terms are simply an enumeration, a simple way of converting pitch intervals to text.

4 Experiment One

4.1 Query Construction

The process for turning a music query into a text query is similar to that of turning a music document into a text document. The query "wrapper", the syntactic sugar, differs for each target system, but the basic method is the same. In our experiments queries are composed of pitch interval unigrams or bigrams. Note that this differs from the Downie, 1999 approach, which uses longer 4, 5 and 6-gram queries as well as 4, 5 and 6-gram document terms. We want to see how well our approach does using these basic units, ignoring the larger discriminatory power (but smaller flexibility and generalizability) that longer n-grams afford.

For Inquery, each of our queries is a weighted sum of either unigrams or bigrams. For Language Modeling, a query is a product of the estimate of each unigram or bigram probability, given the model of the document. Essentially, our query has become a "grab bag", a set of terms. The only sequentiality that has been preserved is that of two contiguous notes (for interval unigrams) or three contiguous notes (for inteval bigrams). For example:

Inquery query	Language Model query
#q1 = #WSUM(1.0	$\hat{p}(Q\|M_d)$ =
1.0 30	$\hat{p}(30\|M_d)$
1.0 32	* $\hat{p}(32\|M_d)$
1.0 16	* $\hat{p}(16\|M_d)$
1.0 34	* $\hat{p}(34\|M_d)$
1.0 18	* $\hat{p}(18\|M_d)$
...	...
);	

Additionally, we construct queries that are somewhere between unigrams and bigrams. Inquery allows *phrase* operators which take as their argument multiple query terms (query content representation nodes) and an integer, x. The uwx (unordered window) operator look for query terms which appear in any order within a size x document window. The odx (ordered window) operator looks for phrase terms which appear in the exact same order as in the phrase operator, within a size x document window.

For example, if the song from Figure 2 is used as a query, it might be transformed into the following weighted sum of Inquery phrase operators:

```
#q1 = #WSUM(1.0
     1.0   #od3(30, 32)
     1.0   #od3(32, 16)
     1.0   #od3(16, 34)
     1.0   #od3(34, 18)
     ...
);
```

We use four different Inquery phrase operators: #uw1, #od5, #od3 and #od1. The purpose of the unordered window (#uw1) is to examine the importance that sequencing has in music and what happens when sequencing is ignored. The purposed of the ordered windows at various lexical distances (#od1, #od3, #od5) is to examine tighter and looser matching schemes, to see to what degree we are helped or harmed by allowing, in effect, non-contiguous bigrams (non-contiguous ordered unigram pairs).

4.2 Results

Thanks to TREC, the text retrieval community has several large collections of documents along with a fairly extensive relevance judgements on each collection (Harman, 1995). The music retrieval community does not yet have this same standardization and document-to-query relevance information. A *name-that-tune*, or known-item search, is an alternative evaluation technique. A song is selected from the collection, a query is created based on that song, and the system is evaluated by the ranking of this item.

Fifty songs were selected at random from the Meldex collection and tranformed into queries of various lengths. Full text queries contain an average of 50 notes (49 interval unigrams, or 48 interval bigrams). 12-note and 7-note incipits are queries constructed by using the initial 12 or 7 notes of a song.

Table 1 lists the Inquery and Language Modeling approaches using unigrams and bigrams, and the Inquery approach using unigram phrases (uw1, od5, od3 and od1). The value shown is the average rank achieved by each retrieval approach over queries from 50 songs. The lower the rank, the better the system performance.

Table 1: Average rank of known item over 50 queries

System	*Query Type*	full-text	12-note incipit	7-note incipit
INQUERY	unigram	259	717	1420
LANGMOD	unigram	1377	1168	1578
INQUERY	bigram	1	14	162
LANGMOD	bigram	1	12	221
INQUERY	Phrase uw1	2155	2916	3731
INQUERY	Phrase od5	22	180	667
INQUERY	Phrase od3	5	98	507
INQUERY	Phrase od1	1	15	164

4.3 Discussion

Query Length One might question the efficacy of using the full text of a known item for query construction. This was done because it gives an upper bound on the performance of each retrieval technique. In text retrieval, if the full text of a document is used to create a query, performance would likely drop. There are too many "noisy" words in a document, words which conceptually have little to do with the *aboutness* of that document. On the other hand, intuition says that the longer a musical passage, the more that passage will be *about* the song from which it is taken. The results in table 1 show that this intuition holds for all these retrieval systems and queries, except Language Modeled unigrams. The larger the size of the query, the better the system performs, on average.

Inquery versus Language Modeling For unigrams, the Inquery approach does much better than the Language Modeling approach. Neither system, however, has an absolute performance level anywhere near acceptable. This is not surprising, since unigrams contain so little sequential information necessary for music. With bigrams, on the other hand, the difference between the two approaches narrows. For full-text queries, Language Modeling does worse than Inquery in only 2 of 50 instances, and equal to Inquery in the remaining 48 instances. It is not currently known why language modeling fails with unigrams, relative to Inquery, but succeeds with bigrams. Unigrams do not discriminate very well between songs, and perhaps Inquery's inverse document frequency weighting handles this problem better than the language modeling approach. A bigram encapsulates a longer sequence of notes than a unigram, increasing implicit term discrimination perhaps to a point where Inquery's weighting scheme no longer provides significant improvement over the Language Modeling scheme.

Inquery Phrases One main difference between the language modeling approach and Inquery is that the former does not contain the notion of phrases, or term non-contiguity, while the latter does. The first phrase experiment, #uw1, performs as expected: poorly. The unordered window size 1 simulates contiguity between unigrams, increasing the discrimination power of the term, but the disregard for the order of the contiguity destroys any discrimination we might have gained. The #uw1 phrases do even worse than unigrams by themselves, on average.

There are exceptions. A query-by-query analysis shows that #uw1 succeeds with (approximately three) queries that have a large proportion of contiguous notes with the same pitch: #uw1(25, 25). In such cases, the first and second term in a phrase are the same, and the symmetry is not broken by an unordered window. A #uw1 phrase effectively behaves like an #od1 phrase, so performance improves. In the vast majority of instances, however, the unigrams are not the same and results are hurt significantly; too many spurious matches are found.

By paying attention to the ordering of contiguous unigrams, recapturing the original sequentiality of a song, *od* phrases provide better performance. The #od1 phrases are extremely similar to bigrams. When the distance between the ordered unigrams is looser (#od3 and #od5), perhaps to allow more flexibility in matching, performance drops; more spurious matches are found. (More relevant songs might be found as well, but we cannot test this without better collections and relevance judgements.) The amount of the drop appears to be inversely proportional to the size of the query. The longer the query, the less the ranking is affected by spurious matches due to larger window sizes.

5 Experiment Two

5.1 Query Construction

We cannot control the length of the user query, but we can control how that query is used. This leads to a second experiment, based on Inquery's flexible phrase operators. (We do not continue with Language Modeling because current text-based approaches do not allow for non-contiguous n-grams, or phrases.)

Recall that Inquery's phrase operators take as their arguments query content representation nodes (sections 2.1 and 4.1). In the previous experiment, these nodes were pitch interval unigrams and bigrams. However, phrase operators are themselves query content representation nodes. Phrase concept nodes can therefore be inferred *from other phrase concept nodes*. Simply put, phrases can be nested.

A sample query with nested phrases is this #od3 phrase of #od5 phrases:

```
#q1 = #WSUM(1.0
    #od3(   #od5(30, 32)
            #od5(32, 16)
            #od5(16, 34)
            #od5(34, 18)
            ...
    )
);
```

An exponential number of phrase combinations and nestings are possible. Phrases can be nested to an arbitrary depth, phrase distance arguments can be varied at arbitrary nodes, indeed the entire shape of the query content node network can be modified and rearranged. For simplicity, we only test #od$x$ phrases of #od$y$ phrases. Trials are run for queries of each length: full-text, 12-note incipits, and 7-note incipits.

5.2 Results

Table 2: Average rank of known item over 50 queries

System	*Query Type*	full-text	12-note incipit	7-note incipit
INQUERY	Phrase od5 of od5	1	9	218
INQUERY	Phrase od5 of od3	1	4	116
INQUERY	Phrase od5 of od1	1	1	18
INQUERY	Phrase od3 of od5	1	2	99
INQUERY	Phrase od3 of od3	1	2	84
INQUERY	Phrase od3 of od1	1	1	13
INQUERY	Phrase od1 of od5	1	1	13
INQUERY	Phrase od1 of od3	1	1	11
INQUERY	Phrase od1 of od1	1	1	8

5.3 Discussion

The results are encouraging for the nested *odx* operators. For any given query at a specified query length, the nested phrase equals or outperforms the unigram, bigram, and straight #od5, #od3, and #od1 forms of that query. The only exception is the #od5 of #od5s on 7-note queries; the phrase windows are too loose and the query length too short. Otherwise, every other query construct outperforms straight #od1 phrases and bigrams.

Query size still matters, but using nested phrase operators in a manner which attempts to recapture the original sequentiality of the song produces more precise results. The advantage is that these looser matching schemes (#od5 of #od3s, and so on) allow potentially better recall without sacrificing too much precision. When doing a known-item search this is not interesting. However, once larger collections and relevance judgements become available (once multiple documents are judged relevant to a single query), the ability to allow for more flexible matching without hurting precision is important. It is the author's opinion that there are benefits in different query formulations and shapes. Different music features and variations could be emphasized or ignored, based on how the query is constructed.

6 Conclusion

The goal of this paper was to evaluate current text information retrieval systems as applied to the monophonic music retrieval task. Although other probabilistic methods such as the Inquery system have been tried, the Language Modeling approach, to the best of the author's knowledge, has never been applied to music. This latter approach did not yield impressive results.

However, the framework offered by Language Modeling is still useful. In most current text retrieval systems, a separate model is used for document represenation and for query-to-document-representation matching. Additional layers of abstraction are added, "concepts" are inferred from documents and from queries, and then these concepts are retrieved. For text, this additional abstraction does not always pose a significant problem. There is often high correlation between tokenized alphanumeric sequences (words) and concepts. With music, on the other hand, concepts are much more difficult to find. Where is meaning found in music? In a single note? In a note interval? In six contiguous note intervals? In six non-contiguous note intervals (i.e.: because of improvisation or some other variation)? The answer remains elusive, and a retrieval system which bases its effectiveness on its ability to extract some sort of musical terms or concepts from a music document will have to solve this problem before tackling the actual retrieval task.

The Language Modeling framework bypasses this intermediate step and goes directly from document to query. It asks how likely it is that a given document could have generated a query, regardless of the "concepts" that are found in either the document or the query. This framework thus becomes very useful for music, where concept extraction is difficult.

Despite these rantings about the difficulty of finding or inferring musical concepts from documents, this paper also shows that an adequate workaround is possible. While other approaches to music indexing have concentrated on longer n-grams, we find that it is possible to index seemingly meaningless terms, pitch interval unigrams, and still get good results. The Inquery phrase operators let a user construct a query which takes into account the ordered distance of unigrams, creating a semblance of the original sequence. A configurable window size as well as the ability to nest phrases to an arbitrary depth allows for flexibility in precision and recall.

However, this Inquery phrase technique is strictly limited to monophonic music. For polyphonic music, it is unclear how one might index interval unigrams. If polyphonic music consisted exclusively of chords, the task would be much simpler. But most interesting music has multiple voices, multiple parallel threads, all of which are not always easily resolved into a single one-dimensional sequence. Other techniques need to be developed.

7 Acknowledgements

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, and NSF grant number IIS-9905842. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor(s).

References

AMNS (2000). Nightingale music notation software. http://www.ngale.com.

Byrd, D. & T. Crawford (2000). Problems of information retrieval in polyphonic music. *Submitted for publication*.

Callan, J., W. Croft, & S. Harding (1992). The inquery retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pp. 78–83.

Downie, J. S. (1999). *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-grams as Text*. Ph. D. thesis, University of Western Ontario.

DT (1996). Digital tradition folk song database. The Digital Tradition. http://web2.xerox.com/digitrad.

Harman, D. (1995). The trec conferences. In R. Kuhlen & M. Rittberger (Eds.), *Hypertext - Information Retrieval - Multimedia; Synergieeffekte Elektronischer Informationssysteme, Proceedings of HIM '95*, pp. 9–28. Universitaetsforlag Konstanz.

Lennon, J. & P. McCartney (1967). Lucy in the sky with diamonds. Recording, from the album "Sgt. Pepper's Lonely Hearts Club Band".

McNab, R. J., L. A. Smith, D. Bainbridge, & I. H. Witten (1997). The new zealand digital library melody index. In *D-Lib Magazine*. Available at: http://www.dlib.org/dlib/may97/meldex/05witten.html.

Ponte, J. M. & W. B. Croft (1998). A language modeling approach to information retrieval. In *Proceedings of ACM SIGIR*, pp. 275–281.

Schaffrath, H. (1995). The essen folksong collection. (Four computer disks containing 6,255 folksong transcriptions and 34-page research guide. http://www.musicog.ohio-state.edu/Huron/publications.html). Center for Computer Assisted Research in the Humanities (Stanford, CA).

Turtle, H. & W. B. Croft (1991). Evaulation of an inference network-based retrieval model. *ACM Transactions on Information Systems 9(3)*, 187–222.