Document Collections

Recall that the first part of our four part model of IR is the document collection. Just what a document is really depends on who defines it. For the end-user the "document" is likely to be a monograph or a set of monographs, a set of articles, or any combination of physical items that can be read, or viewed, or listened to, or studied such that he or she learns – that is, fulfils the information need. To the [traditional] librarian, the document evokes the manuscript – a physical piece of paper. Of course, as you know from other reading, such as Buckland's "What is a document?" or "When is a zebra a zebra?" Nowadays, in computer science, information science, and increasingly in librarianship, a document is any machine stored data record. This includes the source materials, such as word processing documents, which in the end could be used to print a book; likewise any surrogate, article, descriptors, and so on: the document is any computer file, whether or not it is the *original*, a *reference*, or a *surrogate*.

In IR, some authors (e.g., Salton, Allan, & Buckley 1993; Callan, 1994; Mittendorf & Schäuble 1994) see even a paragraph or *any identifiable subsection* as a document consisting as part of a larger whole (the collection). This is an important facet to understand in the role of metadata.

Ultimately we need to match the query to the collection, via the document's computer representation, this is called *mapping*. Specifically how depends on the *model*.

What do we know about the document? Are all documents equally valuable? Should each be stored forever? Which documents or parts of documents should be made available to the user or processed by the IR system? For instance, what might be the best approach for an email IR system? If a book were available online, what would be an appropriate technique for mapping the book-as-document to the query in a way that is both computationally efficient and effective for the user?

Structure of the Document:

All documents have some kind of structure. A typed letter has (probably) a salutation, a date, the return address, the body, a closing, signature, and perhaps a note about attachments. The letter, produced by a word processing program, has other data, too: the metadata created by the word processing program and stored in the document's header. These data are hidden from the user but can be used by the IR program. A set of these let-

ters, say grouped by year, collectively form another kind of structure, here perhaps "correspondence." The concept (topicality, or "aboutness") of correspondence doesn't appear physically; it is a kind of *inherent structure*.

Raw data, too, have some kind of structure. That structure is applied by humans after the data have been collected. The SETI project, for example, isn't looking for *known* data, but instead *any* data from the universe (literally) of potential data. Medical data captured by a medical device by itself has structure, but requires the doctor's interpretation. We see, then, that documents are made useful from a variety of ways: (1) their inherent structure, (2) human (or automatically) applied value, or (3) by combinations of whole, or of parts of, documents into a new idea. [These functions are called *text analysis* or *lexical analysis* and can be performed by humans or automatically by computer with varying degrees of success.]

Before the rise of IR, and of course even today, the emphasis is on how quickly a computer can retrieve files, often measured in nanoseconds. The fastest way to store and retrieve data is to shape it into a structure that is designed for speedy retrieval. This is the underlying idea of *relational database management systems*. The data that are manipulated by the RDMS are not the original but rather a *surrogate*. In libraries, the surrogate was originally the library card; now the *machine-readable catalogue record* (MARC) or some XML data file definition (DFD), or some combination, such as Dublin Core (DC) or MODS. Part of the MARC record includes data that map directly to the item-in-hand (e.g., the book) and part includes data added by the cataloguer to make the record more flexible – that is, meaningful subsets can be made by combining parts of the *structured record* in the surrogate usually bolstered with Boolean combinations. This is how an end-user can search for the title (something extracted from the item) and subject (something added by the librarian). This means that for IR there are many challenges right from the beginning: what document structure will we plan for in our system? Would it be better to have the limited representation of a document offered by the surrogate or some kind of greatly flexible, but computationally and humanly difficult, broader representation of a document offered by parsing the document, by extracting and storing all terms (tokens)?

The Familiar

Librarians are trained in applying standards: name authority standards to harmonize an author's name which facilitates the future clustering of all works by that author, regardless of how the item-in-hand describes him or her. Some standards are numeric: call num-

bers and DOI are unambiguous numeric representations of the document. Call numbers at least suggest to people who know how to create them an immediate knowledge of the item's main intellectual content, the year of its production, the number of copies in the library, and depending on one's skills, a fairly good guess at the author's last name. The DOI is unambiguous, on the other hand, there is no way for anyone to determine anything about the author, the topic, etc., and so it is impossible to use the DOI to create meaningful subsets (clusters) of related documents.

Librarians also create subject heading tracings through a combination of their own knowledge of creating subject headings and the standard, say LCSH, itself. Here is a complete MARC record as an example:

```
LEADER 00000cas  2200000   4500c
001    1481502
005    19831108085205.0
008    750727c19669999dcuar1   y    0ua1a0eng d
010    66025096
022    0066-4200
030    ARISBC
035    0066461|bMULS|aUlp   No.  0827700008|aPITT NO.  0687900000
       |asf68323000|bFULS
040    MUL|cMUL|dFUL|dNSD|dOCL|dDLC|dNSD|dm.c.|dDLC|dSER|dSCL
042    nsdp|alc
049    SCLL[ ][Library][Science][Ref.][Z][699][.A1][A65]
050 0  Z699.A1|bA65
090    Z699.A1|bA65
222 00 Annual review of information science and technology
245 00 Annual review of information science and technology
246 30 ARIST
260    [Washington, etc.]|bAmerican Society for Information
       Science [etc.]
300    v.|c24 cm
362 0  v. 1-   1966-
510 2  Chemical abstracts|x0009-2258
550 1  Vols. 1-2 issued by the American Documentation Institute;
       vols. 3-  by the American Society for Information Science
555    Vols. 1-7, 1966-72.  1 v.; Vols. 1-10, 1966-75.  1 v
570    Editor: 1966-75 C. A. Cuadra
650  0 Information science
650  0 Information storage and retrieval systems
650  0 Libraries|xAutomation
700 11 Cuadra, Carlos A
710 20 American Society for Information Science
710 21 American Documentation Institute
949    standing order lsref acq
```

In this example, the original text is the ARIST volume. The *machine-readable* representation is the MARC record.

Retrieving this record: Why would someone want to read this book? There really isn't an answer because we do not know what current want and future readers will want. Because of this surrogates are made to give maximum combinatory flexibility based on what our understanding of knowledge is, moderated by the historical development of RDMS and computer technology with an eye towards future information needs

How could one retrieve this record? The answer depends both on the design of the IR system – what does the interface let the user see and the what is data structure being searched? Usually online public access catalogues (OPACs) are designed to limit the seeker's view of the data, such as searching only by indexed fields. The fields that are indexed depend on our historical understanding of knowledge. When education was limited to the few, those in the know would refer to a book by its author or its title. As collections of books grew, and as educational opportunities expanded to all society, these common access points became less useful. Subject tracings grew in value. This leads to two main techniques of how end-users find materials: *searching* for known entities, author, title, perhaps subject, and *browsing* for *unknown* entities, usually by subject.

What happens when the surrogate is not created by a librarian or for some other reason there is no standard vocabulary (a *controlled vocabulary*) applied? Obviously, if a MARC record contained a non-standard descriptor, then it would be impossible to co-locate (cluster) all related materials. This leads us to *keywords* and *key phrases*. The term keyword is used too casually and contradictorily. America Online® fosters the idea that people can find whatever they want by inputting the "AOL keyword." In fact, this kind of keyword really is a controlled vocabulary term! Most digital libraries, such as ACM Portal, and most research journals ask authors to assign their own keywords. These keywords do not come from a closed vocabulary, but rather reflect anything the author wishes to include. The rationale is that letting authors assign their own keywords means the terms reflect more closely how people in that research arena express themselves. This isn't a bad idea, but what happens when terms change (and they do!) or when the volume of documents that have the same keyword become too numerous? This leads to another important consideration: in IR we try to identify documents but also try to *discriminate* between documents. Consider a collection of computer science documents. The query keyword "computer" probably will retrieve 100% of the documents! A less common term combined with "computer" into a key phrase will be helpful, but the distinction may be so close or

difficult to compute that there's no real benefit. Because of this, many digital libraries have introduced controlled vocabularies of "keywords" from which authors select their terms.

Thesauri:

A document that uses the same terms repeatedly isn't an interesting read. People write to be read and so generally, and one hopes, they infuse their writings with elements of style. Stylistic and rhetorical devices are informative, but also confound an IR system. To say, as Little Big Man's grandfather does in the movie of the same name, "My heart soars like an eagle" makes no sense in IR nor in machine translation (MT). In IR, the sentence becomes the set of terms {"heart", "soar", "eagle"} which will match tokens from a document but certainly doesn't convey the speaker's intended meaning, happiness. In MT, this phrase is nonsense because of the way MT parses language. Hearts do not have the property of flight, so "heart soar" is nonsense. Humans, and perhaps some MT systems, can determine when a term is roughly equivalent to another term: "cat" and "kitty"; "dog" and "canine", and so on. Of course, a kitty is a minor cat; a "canine" is a more formal appellation of a dog. Some IR systems can track this linguistic difference; most don't. What librarians know is that when searching people can broaden the search by using less semantically-stringent terms and narrow the search by using more specific terms. As a result in IR, an implementation could be programmed to *conflate* automatically (or *map*) the user's query with terms from a thesaurus. The user inputs "cat" and the IR system searches for "cat" + "tomcat" + "pussy" + "kitty" + "felix Americanus", etc. The purpose of this conflation is an attempt at introducing some of the pragmatic elements of language use into IR.

Thesauri might also keep track of *homonyms*, terms with identical spellings but different meanings. The "post" can mean the mail, but also the piece of wood and the act of hanging an ad on the wall. *Tomcatting* around may derive from the behavior of the cat, but is not the same as "tomcat" and IR systems do not keep track of the syntactic differences between the two tokens (one's a gerund, the other is a noun).

Finally, some thesauri are used that track when a term appears with another term (*co-occurrence*). There are many ways of doing this: special list of terms (like a stop list) that are searched for in new documents or automatically generating a co-occurrence table of every term mapped to every other term.

Thesauri are used in *pre-coördination* activities; that is preparing documents before the public uses them by adding surrogates and descriptors. Library thesauri standardize the

use of terms by human indexers to assign standard terms and association, to improve retrieval and permit collocation. This example is from Inspec:

```
Computer-aided instruction
      See also education
      UF teaching machines
      BT educational computing
      TT computer applications
      RT education
      RT teaching
```

Thesauri can be applied also automatically. Here a computer file holds terms that are divided into thesaurus classes. Similar terms from one class are replaced by terms from another class (Salton and McGill):

408	Dislocation Junction Minority-carrier n-p-n p-n-p point-contact recombine transition unijunction	409 410	Blast-cooled Heat-flow Heat-transfer Anneal stain

Thesauri are specific to a subject area and so contain terms that make sense or are important within a field. To a degree, this addresses the issue of semantic ambiguity and semantic interoperability. In IR the goal is to create thesauri classes of terms of "moderate frequency" – ideally, the classes should have similar frequency. Determining frequency varies by the designer's model of language. Lundgren, for example, created linguistic distribution of all terms in Swedish that make it possible to calculate chi-square for all terms, which demonstrates the multimodal distribution of terms in a language. Most English-language researchers, however, use a *linguistic corpus*, such as the one available from Brown University (http://clwww.essex.ac.uk/w3c/corpus_ling/content/corpora/list/private/brown/brown.html) or other *linguistic corpora*. See http://www2.lib.udel.edu/subj/ling/internet.htm; http://www.tc.columbia.edu/academic/tesol/Han/corpora.html

Here are some examples of subject-specific thesauri.

Alexandria Project

```
canals
A feature type category for places such as the Erie Canal.
Used for:
The category canals is used instead of any of the following.
    canal bends      canalized streams      ditch mouths
```

```
    ditches          drainage canals        drainage ditches      ...
more ...
Broader Terms:
Canals is a sub-type of hydrographic structures.
canals (continued)
Related Terms:
The following is a list of other categories related to canals (non-
hierarchial relationships).
    channels         locks      transportation features     tunnels
Scope Note:
Manmade waterway used by watercraft or for drainage, irrigation, min-
ing, or water power. »   Definition of canals.
```

Getty Trust Art and Architecture Thesaurus

- Controlled vocabulary for describing and retrieving information:
- fine art, architecture, decorative art, and material culture.
- Almost 120,000 terms for objects, textual materials, images, architecture and culture from all periods and all cultures.
- Used by archives, museums, and libraries to describe items in their collections.
- Used to search for materials.
- Used by computer programs, for information retrieval, and natural language processing.

Provides the terminology for objects, and the vocabulary necessary to describe them, such as style, period, shape, color, construction, or use, and scholarly concepts, such as theories, or criticism.

Concept:

a cluster of terms, one of which is established as the preferred term, or descriptor.

Categories:

associated concepts, *physical attributes*, *styles and periods*, *agents*, *activities*, *materials*, and *objects*.

Here is a record applying the Art& Architecture Thesaurus:

Record ID: 198841
Descriptor: rhyta
Note: Refers to vessels from Ancient Greece, eastern Europe, or the Middle East that typically have a closed form with two openings, one at the top for filling and one at the base so that liquid could stream out. They are often in the shape of a horn or an animal's head, and were typically used as a drinking cup or for pouring wine into another vessel.
Hierarchy:

```
Containers [TQ]
...<containers by function or context>
..........<culinary containers>
```

```
      ..................<containers for serving and consuming food>
Terms:
      rhyta
      rhyton (alternate, singular)
      protomai
      protome
      rhea
      rheon
      rheons
Related concepts:
      stirrup cups
      sturzbechers
      drinking vessels
      ceremonial vessels
```

The following section describes in brief some of the influences on thesaurus construction.

Automatic Thesaurus Construction

Approach

- Select a subject domain.
- Choose a corpus of documents that cover the domain.
- Create vocabulary by extracting terms, normalization, precoordination of phrases, etc.
- Devise a measure of similarity between terms and thesaurus classes.
- Cluster terms into thesaurus classes, using complete linkage or other cluster method that generates compact clusters.

Normalization rules map variant forms into base expressions. Typical normalization rules for manual thesaurus construction are:
(a) Nouns only, *or* nouns and noun phrases.
Singular nouns only.
Spelling (e.g., U.S.).
Capitalization, punctuation (e.g., hyphens), initials (e.g., IBM), abbreviations (e.g., Mr.).
Usually, many possible decisions can be made, but they should be followed consistently.
Which of these can be carried out automatically with reasonable accuracy?

Terms to include:

- Only terms that are likely to be of interest for content identification.
- High-frequency terms should be ignored (large stop-list).
- Ambiguous terms should be coded for the senses likely to be important in the document collection.
- Each thesaurus class should have approximately the same frequency of occurrence.
- Terms of negative discrimination should be eliminated.

Documents about the document: Imagine the day of books of library catalogues, such as the *National Union Catalog*, that were the primary means of locating items. These books formed the core of a university library's reference collection. Such retrieval aids

books consisted of copies of millions of library cards. The library card's data are not especially useful for indicating the contents of the book, other than the subject tracing, but were a boon to locating a copy of a desired monograph or even just to learn of its existence. From this limitation arose the idea of *abstracts*, *reviews*, and *extracts*. These are printed, informative, textual descriptions of the content and, of course, themselves form a document. Given a RDMS that searches library cards and some other kind of system that tells the end-user *about* the contents of the book, which would you recommend? The RDMS might be very fast in creating lists of references. This other system might be slower and perhaps error-prone, but the results are more useful to the information seeker. At one time, these were the only options. Now, as you'll see, the concept of document is changing such that both the original's content and descriptive data about the content can be integrated into new forms of document structures to combine the power of RDMS and the flexibility of IR models.

First, however, we must consider other types of documents (specifically surrogates for non-textual materials) and then how *indices*, (*indexes*), or *inverted files*, are created.

Additional note: We won't discuss these items here, but you may be interested in reading about fine-grain processing (comparing ASCII and Unicode), data compression techniques, such as doubly-linked lists, hash tables, and enumerations. The specifics of increased storage efficiency is part of CS.

Text documents:

All text documents to the computer appear as strings of codes. These codes vary by locale: for us, the code is ASCII, although increasingly the representation is in Unicode (ISO-8859-1, aka Latin-1). These encoding schema identify which are alphabet letters (lower case and upper case) [a-z, A-Z], which are numerals (0..9), and which are specifically computer codes (those intended to be "seen" only by the computer, such as "backspace"), and those that are "white space." White space characters include the space (the code that create the space between words) and the codes for tab (\t), vertical tab (\v), carriage return (\r) and "new line" (\n). Strings of codes can also be meaningful to humans. These are "words", aka "tokens", aka "semantic units." It doesn't make sense to parse the document down to the level of the individual character code, but rather good sense to make an computer file, the index or inverted file, that stores the word (aka the *vocabulary*) and

the number of times the word occurs (*occurrences* or *frequencies*). [See Baeza-Yates, p. 192ff.]

Optionally, during the parse, the language model may ignore words stored in *stop lists* (called *stop words*). As you know, these are words which are syntactically important but are ignored as having little linguistic (!) importance or which are not good discriminators. The token "the" for example is probably the most commonly-occurring term in English; a query that includes the term "the" would retrieve *every* document.

Stop lists:

Stop lists are used to remove terms that are too "noisy" – that is, they act neither as good discriminators or as useful terms to be matched. Kucera and Francis (1967) write that the words "the" and "of" account for 10% of word occurrences in English. "And", "to", "a", and "in" account for the next 10%. In brief, about 250-300 of the most commonly occurring words in English appear in about 50% of any given text. Therefore techniques have been created to refine retrieval. One technique is called *term weighting*, which we cover in the section on ranking. Another technique is to remove the terms at the beginning of the IR process, hence the stop list, which is also called a *negative dictionary*.

Often a concept is expressed not through a single term, but through a phrase. In that case, then, the creator of the stop list must decide whether or not to include both terms in the list. One solution is the search for term *A* and then for term *B* and then retrieve the documents for the intersection of the two. Another solution is to use a *phrase list* (e.g., Benoit, 2002) to tell the parser to ignore the stop list's instructions and keep the phrase intact.

Stemming:

Uncontrolled vocabularies (the kind of keywords above) can have many similar terms for a shared concept. Each of these terms (computer, computers, computing, compute, computes, computed, computational, computationally, computability, computable) reflect some aspect of "computing." A stemming algorithm, such as the Porter stemming algorithm used for the English language, reduces each of those tokens to a base form: "comput".

Some languages, such as Russian, employ many *prefixes* to verb roots to shade the meaning of a term; a Russian parser, then, might want to stem prefixes as well as suffixes.

Stemming in other languages might be easier. In Arabic, for example, words are written usually without vowels and some other grammatical elements, leaving the context

of the term for discrimination. The letters "ktbr" can mean "to write", "writing", "scripture", "written" and so on. Other languages, such as Chinese, make tremendous use of prefixes that would destroy the meaning of the word if removed. The Chinese character for "good" is a combination of the character for "mother" and "child". The concept "good" would be completely lost if the two were separated.

The Caucasian languages, Turkish and related tongues, and most American Indian languages are agglutinative languages, in which syntactic relationships may be expressed by distinct suffixes. This results in a complex morphology that cannot be stemmed as in English.

German has "separable verbs," verbs whose meaning depends on two parts of the verb appearing in different parts of the sentence. For instance, "Wachen Sie auf!" means "Wake up!" in the imperative, but the infinitive is "aufwachen."

Finally, there are differences within a language. The British, Australian and (often) Canadian spellings of words can differ from American spellings "colour" vs. "color") and there exist variants ("spanner" vs. "wrench").

Creating an index:

Here is a small example. Note that here we started counting from 1 and add one to the count between words for the blank space code. Note, too, that the third row shows only the words included in the index if we used a stop list. The fourth row shows the terms after *stemming*:

"To get to Newburyport, take the MBTA from Boston's North Station."

1	4	8	11	22	27	31	36	41	50	56
To	get	To	Newburyport,	take	the	MBTA	from	Boston's	North	Station.
			Newburyport			MBTA		Boston's	North	Station
			Newburyport			MBTA		Boston	North	Station

If these terms formed our index and we searched for "North", the IR system could collect letters ("read everything until a white space") to create a word, then compare the word to our query. If there were a match, the system would retrieve this record. This type of sequential search would not be effective or efficient because *every* term would have to be inspected. Note, tho, we have the basics for *binary retrieval*: the query term appears in the document or it doesn't.

Now our index:

Vocabulary	Occurrences (Frequency)

To	2
Get	1
Newburyport	1
Take	1
The	1
MBTA	1
From	1
Boston	1
North	1
Station	1

Of course, this example is both useless and inaccurate. A real document collection may require a large index where only stemmed, non-stop-list terms are stored, along with their frequency, normalized frequency, source document IDs, and, depending on the model, links to other documents or terms for *proximity* searching. An IR system can use any kind of data in the index: there's no reason why a researcher might not want to store the *probability* of a term's appearance given the entirety of the collection, or given the probability of another term's membership in the document. This gives rise to ideas of *term collocation* and to *probabilistic retrieval*. Here is a revised index with more documents (not shown):

Vocabulary	Frequency	Location (document numbers)
Newburyport	3	3, 49, 102
MBTA	29	3, 49, 59, 100, 101, 109, 198…
Boston	1029	1, 2, 3, 4, 5, … 49, 50, 51 … 1029
[And so on…]		

This type of index is useful, but not as computationally efficient as it could be. To improve the index's physical construction (and hence its use later during retrieval), there are techniques to save physical space (called *block addressing*), vocabulary tries, suffix trees and suffix arrays, signature files and others. See Frakes & Baeza-Yates, 1999 or van Rijsbergen, 1997, chapter 4.

For our purposes, the index provides data that reflects the frequency of terms (1) within documents and (2) across the entire collection. The assumption in IR is that frequency of a term (and by extension concepts that use the term or a synonym) reflects the content or "aboutness" of the document. [There are some techniques that are based on the data (the terms) being independent; others on their being dependent. As you suspect, there are different formulae used depending on this fact. In addition, this is the basis of Zipf's "law" and many other theories; but whether this is empirically true or merely a statistical coincidence is an open question.] The goal is to make these data useful at (a) retrieving documents and (b) ranking retrieved documents. [We return to this issue under similarity measures.]

To make the index useful at retrieving documents, the query and the index contents must be mapped to each other. This is discussed later in the section on Framework.

Keep in mind that full-text processing creates indices that ignore punctuation and paragraphs. Most indices also use stemming.

Text documents and SGML:

SGML, standard generalized markup language, is the model of all XMLs. HTML is the most commonly encountered, but these tags provide no contextualization, e.g., <b> means "bold" which alters the *display* of the data on the screen or printout but provides no knowledge about the data. XML implementations, such as MODS and EAD, may be most commonly known in librarianship. The purpose being SGML is to integrate into *text* documents the kind of useful structure found in *database tables*. For example, MARC has a field for "personal name author" (110 field); a full-text document has nothing specifically indicating within the text the author's name. Adding an appropriate SGML tag (say from your own digital library's *xml schema* or DTD, such as `<author>`), it's possible to combine the flexibility of full-text retrieval with the utility of database-styled matching. Moreover, SGML tags can be used to add value to the full-text document, integrating the inherent structure mentioned above. For example, a research paper might have a section identified in the paper as "Results." The designer of the IR system might want to preserve this sectional information and store it in the index. An SGML-marked-up document could have finer grain segmentation: the paragraph with the most significant research results could be identified, perhaps `<results value="primary">` or something to the effect integrated into the text and likewise preserved in the index. If so, then users could search for any term in the document; or any term within the results section, or terms which the author (and information seeker) consider "significant". One can easily imagine an interface permitting an end-user to submit a query term and somehow indicate the value of that term, such as "return the most important or primary results."

Non-text documents:

All non-text documents can be represented in the document collection through their surrogates. [There are other techniques for searching documents without using a surrogate, such as extracting some inherent feature, such as color saturation. This is discussed later.]

For the moment, consider only surrogates used for non-text documents. A photograph in a photographic archive has a weak equivalent of the MARC record above.

Title: Majnun in the Wilderness, folio from an album
Item Identifier: 1951.103 (Accession Number)
Work Type: Painting with **Calligraphy**
Creator: Ruhani
Production: Iran
Dimensions: 40.5 cm x 29.6 cm actual
Nationality/Culture: Persian; Persian
Materials/Techniques: Ink and opaque watercolor
Note: Gift of Gordon Washburn
Repository: Arthur M. Sackler Museum, Harvard University Art Museums

Record Identifier: HUAM100301

[http://via.harvard.edu:9080/via/deliver/fullRecordDisplay?_collection=via&inoID=32662&recordNumber=44&method=view&recordViewFormat=list]

GIS: Geographic Information Systems:

GIS documents are another type of source. To create an IR system for GIS documents, start with the usual parse to extract descriptive language that surrounds the document, as well as data that is inherent to that type of document, such as longitude and latitude, USGS data, and, importantly, the raw matrix data that is read by GIS computer applications to make 2D and 3D images.

Music and Sound:

Music and sound documents, like all other non-textual documents, have surrogate data, as well as data within the source. For example, one could search for "Bach's Toccata and Fugue in D minor" and retrieve the following MARC record:

```
LEADER 00000cgm  2200000Ia 4500c
001    24812285
005    19920629084705.0
008    911119s1991    cau120            vaeng d
040    CIA|cCIA|dOCL|dTKN|dIOM|dOCL
049    SCLL
099    Video|aDisc|a18
```

```
245 00 Fantasia|h[videorecording] /|cWalt Disney Productions, Inc
260    Burbank, CA :|bWalt Disney Home Video,|c1991
300    3 videodiscs (120 min.) :|bsd., col. ;|c12 in
500    Deluxe Cav Laserdisc
500    Commemorative program ([29] p. : col ill.), certificate of
       authenticity and Chapter index, laid in container
500    Title on container: Walt Disney's masterpiece Fantasia
500    A videorecording re-release of the 1940 film
505 0  Toccata and fugue in D minor / J. S. Bach. -- The
       nutcracker Suite / P. I. Tchaikovsky. -- The sorcerer's
       apprentice / P. Dukas. -- Rite of spring / I. Stravinsky.
       -- The Pastoral symphony (The sixth) / L. Beethoven --
       Dance of the hours / A. Ponchielli. -- Night on Bald
       Mountain / M. Moussorgsky. -- Ave Maria / F. Schubert
508    Sound restoration, Terry Porter; technical manager, film
       restoration, Leon Briggs
511 1  Leopold Stokowski, conductor; Philadelphia Orchestra;
       Deems Taylor, narrator
650  0 Animated films
700 11 Stokowski, Leopold,|d1882-1977.|4cnd
700 11 Taylor, Deems,|d1885-1966.|4nrt
700 11 Bach, Johann Sebastian,|d1685-1750.|tToccatas,|morgan,
       |nSuite.|nBWV 565,|rD minor
700 11 Tchaikovsky, Peter Ilich,|d1840-1893.|tShchelkunchik.
       |pSuite
700 11 Dukas, Paul,|d1865-1935.|tL'apprenti sorcier
700 11 Stravinsky, Igor,|d1882-1971.|tLe sacre du printemps
700 11 Beethoven, Ludwig van,|d1770-1827.|tSymphonies,|nno. 6,
       op. 68,|rF major
700 11 Ponchielli, Amilcare,|d1834-1886.|tLa Gioconda. Danza
       delle ore
700 11 Mussorgsky, Modest Petrovich,|d1839-1881.|tUne nuit sur le
       mont Chauve
700 11 Schubert, Franz,|d1797-1828.|tEllens Gesang,|nD. 838
710 21 Walt Disney Productions
740 01 Sorcerer's apprentice
740 01 Nutcracker suite
740 01 World is born
740 01 Pastoral symphony
740 01 Dance of the hours
740 01 Night on bald mountain
740 01 Ave Maria
740 01 Walt Disney's masterpiece Fantasia
```

Of course, so far retrieval of the music is through the text surrogate. This particular example demonstrates how much non-relevant material can be retrieved along with the relevant.

Example. The inherent aspects of music or sound include pitch, duration, notes, and so on. IR includes techniques that match these aspects. A traditional tool in music librarianship is a printed catalogue of tunes, called a concordance, where the themes are rewritten in the key of C. The information seeker looks for the second note of the phrase but must be able to read music. Say the piece of music is Bach's famous "little fugue" in g minor, BWV578. The first three notes are g – d – b flat. The first two are quarter notes, the last is a half-note. Searching for this piece of music the traditional way using a printed collection of all Bach's works, called a *concordance*, requires the information seeker to transpose the theme to the key of C major. The first note, the g, becomes C. Next the d becomes g: four whole notes distant. The third note (b flat) becomes e, two whole tones

down. You can see that there is a recognizable pattern, but the pattern must be put into a standard (here, the key of C). Since the music notes can be converted to text, music retrieval becomes a text-styled IR question.

Imagine a computer program where you input the notes themselves in the original key signature, using an electronic piano's keyboard and a MIDI interface. The sound that is input along with the stored (sampled) sound are represented by code (here frequency in Mhz) and the distance between any given notes can be calculated. This is exactly how non-binary retrieval operates! [This leads to the idea of "nearest neighbor" searching, something we review later.]