Browsing Mixed Structured and Unstructured Data

Information Processing & Management
42 (2), March 2006, pages 440–452.*

Robert M. Losee
Manning Hall, CB#3360
U. of North Carolina-Chapel Hill
Chapel Hill, NC 27599-3360

losee@unc.edu

October 3, 2005

Abstract

Both structured and unstructured data, as well as structured data representing several different types of tuples, may be integrated into a single list for browsing or retrieval. Data may be arranged in the Gray code order of the features and metadata, producing optimal ordering for browsing. We provide several metrics for evaluating the performance of systems supporting browsing, given some constraints. Metadata and indexing terms are used for sorting keys and attributes for structured data, as well as for semi-structured or unstructured documents, images, media, etc. Economic and information theoretic models are suggested that enable the ordering to adapt to user preferences. Different relational structures and unstructured data may be integrated into a single, optimal ordering for browsing or for displaying tables in digital libraries, database management systems, or information retrieval systems. Adaptive displays of data are discussed.

*The author wishes to acknowledge the useful, constructive comments of an anonymous referee.

1 Introduction

Browsing an adaptive database system with both structured and unstructured data can be achieved through simplifying data structures and using subject carrying indexing information to order the data. The indexing terms or metadata may be assigned probabilities, costs, and benefits, and a system then may adapt its internal organization and its output to these user-based probabilities or costs, as well as to other metaphors (e.g., information theoretic) for user needs and interests. Information retrieval systems have been developed that can manipulate structured data, as have relational and object oriented database systems that manipulate text fragments or multimedia (Borkar et al., 2001; Sabin and Yap, 1998; Vasanthakumar et al., 1996; Yan and Annevelink, 1994), as well as XML encoded media (Blanken et al., 2003). Our system integrates the information in a structure that can be optimally ordered for browsing, regardless of the type of source and the type of data, e.g., structured, semi-structured, or unstructured.

Facts may be retrieved and presented consistent with minimizing the dissimilarity between adjacent facts, as well as the degree to which the facts match with the query. If we conceptualize the output of a structured database as a table of rows and columns, a system can learn the characteristics of rows and columns of interest of a user or group of people and display a table consistent with the user's or users' preferences. Similarly, if the user wishes to retrieve documents or other media, the data can be arranged to reflect the user's expressed interests as a table for browsing or retrieval. Our purpose here is to integrate these structured and unstructured models to provide integrated output that might contain, for example, factual numeric data about the population and average temperature for a country, along with a picture of the capitol building, an audio recording of the country's national anthem, and a journal article about that country's economy.

In the case of an adaptive database system with stored information, the organization of the data for ultimate use by a query producer is the primary means of optimizing the data for a given user or group of users. By allowing systems to adapt to these user-preference-based aspects of structured, semi-structured, or unstructured data, a system may improve the user's experience in a variety of ways. *We integrate data for the user, with data being organized and presented based on the user's needs, not on the type of database structure used to store the information, e.g. a relational database or a set of HTML web pages.*

2 Representing Data

Surrogates for documents or facts may be consistent with any of a number of representational methods (Cover and Thomas, 1991; Hamming, 1986; Losee, 1990). Assigning indexing representations and metadata features may have several inconsistent goals: to best represent the author's intent; to best represent the needs of the individual user or a group of users; or to best represent what is most special or unique about the data (Losee, 1998).

The representations of data may be viewed as *structured*, *semi-structured*, or *unstructured*. Structured data is organized in a highly regular way, such as in tables and relations, where the regularities apply to all the data in a particular dataset. Semi-structured data would contain this same information, but instead of having regular structures applying to all items in the dataset, data might be interpreted with structural information supplied as tags, such as *name="Bob"*, *city="Chapel Hill"*, *state="North Carolina"*. In such a case, one can move the name, city, and state components around as one moves from information about one person to another. The structural regularity across data items is gone. Missing data may be represented by the presence of a label with a null attribute or an attribute indicating "missing", or the label may be omitted completely. Unstructured data, such as text or images, contain information but contain no explicit structuring information, such as tags. However, these tags may be assigned using manual or automatic techniques, converting the unstructured data to semi-structured data.

Tags representing index terms or metadata may be assigned by humans, or may be produced or selected through automated procedures. The term *metadata* is often applied to index features that are part of a formal indexing system that is meant to be used beyond a single academic or institutional environment. Metadata is a more recent term than indexing, being applied most frequently to recent systems for representing characteristics of entities, often based on the Dublin Core or RDF (Resource Description Framework) (Greenberg, 2002; McCray et al., 1999; Moens, 2000; Salminen et al., 1995).

The metadata used to describe a topic, fact, relation, or an entire document may be produced from combining the individual metadata items assigned to specific features. Index terms or metadata may be arranged a number of ways to represent a single topic. A *simple linear list* includes all index terms and can be used to indicate whether the data being indexed *is* or *is not* about the topic in question.

A different, *hierarchical*, arrangement is used in many topical classification systems. If the books in a library were divided into history and non-history books, with *history* being the primary binary feature, the second feature for history books might represent something historically specific, such as pre- or post-1950, or northern or

southern hemispheres. The second feature for non-history books may be something else, such as social science vs. non-social sciences. Libraries usually use hierarchical classification systems with context-sensitive feature meanings. This hierarchical method is used in CART (Duda et al., 2001). Each feature may represent a different level in a decision or regression tree or the output from a similar technique, with many of our features having multiple semantic values corresponding to the set of features at a particular regression tree level. In the case where meaning depends upon context, the probabilistic feature ordering techniques described below will perform at a lower level than would occur if all features were unambiguous and their values were independent of their context. User preferences may be treated as factors (e.g. economic) in the feature selection algorithms within procedures such as regression trees, producing user oriented classification and ordering.

For applications below, we assume a simple linear arrangement (Gaede and Gunther, 1998; Jagadish, 1990), or feature vector, for browsing (Cover and Walsh, 1988; Hearst et al., 2002; Losee, 1992; Losee, 1993; Losee, 1997a; Morse, 1970) or retrieval (Kowalski, 1998; Losee, 1998) purposes. In most cases, the presence or absence of a feature is indicated in the representation with a 1 or a 0. This may be enhanced by using statistical techniques such as Latent Semantic Indexing (LSI) (Deerwester et al., 1990; Hull, 1994; Manning and Schutze, 1999) to reduce the dimensionality of the feature space and to produce statistically independent features. A wide range of systems have been developed that assume binary independent features and that perform successfully (Lewis, 1998), and we continue with this assumption. A system using all available characteristics is expected to outperform a system using a limited set of human developed (and more intuitive) index terms or metadata.

3 Representing Facts

Representing and organizing data is an important factor in effectively storing and retrieving information. The choice of an organizing principle that supports both structured and unstructured data and that also inherently adapts to user preferences will allow for the development of a more general model for data storage systems.

Attributes can represent values from a domain of all possible values. Attributes and key attributes, those attributes that serve as the entry points to relations, are described more fully in traditional database and information retrieval texts, to which the reader is referred if they are unfamiliar with these topics. We assume that structured and semi-structured data is organized for our purposes consistent with best practice in relational manipulation, so that there are no insertion, update, or deletion anomalies occur, and the relations are fully normalized (Arenas and Libkin, 2003; Lee, 1987).

Such anomalies would have little direct impact upon browsing, but are considered desirable features in database design for management purposes. The availability of properly normalized data would make it much easier to extract key-attribute pairs for ordering below.

We denote a *minimal fact* as the representation of the relationship between two informational representations, one referred to as the key, $\mathcal{K}$, and the other referred to as the attribute, $\mathcal{A}$. The key is the representation of an attribute about whose referent the non-key attribute provides information (Losee, 1997b). The key and the attribute are representations, and themselves may be described or represented by indexing or metadata. The metadata representations for the key and attribute are denoted as $\mathcal{M}_\mathcal{K}$ and $\mathcal{M}_\mathcal{A}$, respectively. Each metadata set $\mathcal{M}$ contains a vector of n individual metadata items m_1 through m_n: $\mathcal{M} = \{m_1, m_2, \ldots, m_{n-1}, m_n\}$. These are the indexing features used in our description below; all are assumed to be binary for this analysis, although more complex systems are available for non-binary representations (Losee, 2003).

The representation or code for a *factual representation* is the 5-tuple

$$\mathcal{F} = \{\mathcal{M}_\mathcal{U}, \mathcal{M}_\mathcal{K}, \mathcal{K}, \mathcal{M}_\mathcal{A}, \mathcal{A}\};$$

the metadata associated with the union of the metadata for the relation, $\mathcal{M}_\mathcal{U}$; the metadata associated with the key, $\mathcal{M}_\mathcal{K}$; the key's value, $\mathcal{K}$; the metadata associated with the attribute, $\mathcal{M}_\mathcal{A}$; and the attribute's value, $\mathcal{A}$. The structure of a factual representation contains:

- the Gray code for the logical union of the relation's metadata vectors (key and attribute),
- the Gray code for the factual representation's key vector,
- the key (often text),
- the Gray code for the factual representation's attribute vector,
- the factual representation attribute (often text).

When we refer to the Gray code for an application, we refer to the binary representation of the metadata vector, e.g., the presence or absence of each feature, with the Gray code providing an ordering for the vectors. This ordering is discussed below and examples are given in the next section. The proposed ordering provides a different and superior ordering for browsing purposes than that obtained with traditional binary ordering (e.g., 00, 01, 10, 11).

4 Organizing Data as Factual Representations for Browsing and Display Applications

Using the structures suggested above for factual representations, we may place existing structured and unstructured data into a common data structure suitable for integrated browsing and retrieval.

Structured data may be easily represented using factual representations, although complex relations may need to be decomposed into a series of $< key,\ attribute >$ 2-tuples. Possibly the simplest data structure for structured data consists of a 2-tuple containing a single key and a single attribute. For example, cats are of particular type, suggesting the relationship $(cat\_identifier, cat\_type)$, with one instantiation being $(Tigger, Tortie)$. Here, the cat identifier is the key for the relation, which is used to access the relation, while the attribute (in this instance, of *Tigger*) is *Tortie*.

When storing a structured relation as a factual representation, it might be stored as before with the metadata for the key, the metadata for the non-key attribute, and the union of these two metadata sets in a factual representation. The 5-tuple relation described above might then appear as {*(cat, identifier, type), (cat, identifier), Tigger, (cat, type), Tortie*}, where sets of metadata are shown here in parentheses.

In the case where there are multiple attributes for a single key, the relationship between each attribute and the key is stored as a separate factual representation. In the case of a compound attribute, a synthetic and unique identifier is placed as the attribute, and new compound facts are developed so that the unique identifier is the key for each attribute in the original compound attribute in the new factual representations. Ideally, these unique identifiers shouldn't be presented on a display, being replaced with the compound attributes, or placed into a fuller version of the relation, as described later in this paper.

As an example, consider the case where Caitlyn is the key and her father's first name is Bob and his last name is Losee. The compound attributes could be represented as two different compound facts: that Caitlyn has a father whose first name is Bob and a second fact, that Caitlyn has a father whose surname is Losee. The preferred method here is to assign a unique identifier to Bob Losee and indicate that Caitlyn has father 12345, while there are two relationships, one indicating that father 12345 has the first name of Bob and a second relation indicating that father 12345 has the surname of Losee. One can also represent this unique identifier for notational purposes by combining the attribute fields, e.g., using *Bob_Losee* instead of 12345. We may find the following factual representations:

- { (name, daughter, father, firstname, surname), (name, daughter), Caitlyn, (name, father, firstname, surname), Bob_Losee }

- { (name, father, firstname, surname), (name, father, firstname, surname), Bob_Losee, (name, father, firstname), Bob }
- { (name, father, firstname, surname), (name, father, firstname, surname), Bob_Losee, (name, father, surname), Losee }

Unstructured data are stored with the image, document text, or other binary large objects ("blobs") being a single large attribute, and the metadata associated with the attribute being derived automatically or manually from the attribute. Other attributes about the document, such as title, author, etc., can be treated as additional attributes in added factual representations with the same key as the factual representation containing the document.

Consider an audio file of cat howls stored at a particular URL. The factual representation for this would contain the following: { *(url,cat, howls, mp3), (url), "http://cat.howls.us", (cat, howls, mp3), [audio file here]* }.

We propose that structured, semi-structured, and unstructured data be converted into factual representations and then organized using the Gray code so as to achieve an interleaved ordering of data of all original types.

5 Ordering Metadata with the Gray Code

Information and factual representations may be represented using binary forms, as discussed above. The information may then be sorted using any of a number of techniques. The Gray code to be discussed here is a binary representation system: we use it because it has several desirable techniques for ordering data.

Ordering consistent with the Gray code allows binary data to be placed near similar data (Faloutsos, 1988; Losee, 1992; Losee, 1997a). When the traditional Gray code is combined with probabilistic or economic conditions, adaptive systems may be developed consistent with particular circumstances, reflecting particular characteristics of users, as well as specific utilities for a user or group of users.

The binary Gray code is an ordering system by which, when counting, one number's representation differs from the next number's representation (adjacent to it) by one bit position. When using the binary Gray code (all Gray codes are assumed below to be binary codes to simplify discussion), the Hamming distance, or number of bits by which two adjacent representations differ, is always one when enumerating. Similarly, when ordering existing data using the Gray code, the distance between adjacent representations will be 1 when all possible representations are present once. When there are missing representations, ordering by the Gray code may not be optimal (although it is still likely to be highly effective and may still be optimal).

True Number	*Decimal Representation*	*Binary Representation*	*Gray Code Representation*
0	0	000	000
1	1	001	001
2	2	010	011
3	3	011	010
4	4	100	110
5	5	101	111
6	6	110	101
7	7	111	100

Table 1: Decimal numbers, "regular" binary numbers, and binary Gray coded numbers.

Considering Table 1, we can see that the Gray code does allow for each representation to differ from its immediate neighbors by one position. Counting using the Gray code can be envisioned as moving from 0 to 1 for the first two lines of Table 1, and then placing a 1 in the second (the "two"s) column as the Gray code representation for the decimal numbers 2 and 3 *reflect* (as would a mirror) the Gray code representation for the decimal numbers 0 and 1. The Gray code representation for 4 through 7 then reflect the numbers for 0 through 3 with a 1 in the "four's" column, and so forth. This reflected Gray code is the most commonly discussed Gray code, but there may be times when a non-reflective Gray code may be useful (Losee, 2002). An example of a non-binary non-reflective Gray code is provided in the following order for Base 3 numbers: (00, 01, 02, 22, 21, 11, 12, 10, 20).

Factual representations may be ordered by using the Gray code. The key used in sorting is the concatenation of the data as presented in the structure of a factual representation: $\{\mathcal{M}_{\mathcal{U}}, \mathcal{M}_{\mathcal{K}}, \mathcal{K}, \mathcal{M}_{\mathcal{A}}, \mathcal{A}\}$. This has the effect of placing all 2-tuples of the same type, i.e., with the same metadata, together. Then, within tuples of the same type, those with the same metadata for the key and the same key value are placed together. Then, among the $\mathcal{F}$s of the same type and with the same key, the 5-tuples are ordered so that similar types of attributes are placed together, and then within those, 5-tuples are ordered by the value of the attribute.

Once the factual representations have been ordered this way, users may browse through data, both integrated structured, semi-structured and unstructured data, by beginning at a starting point, usually the point that is most relevant or closest to the original information need or expressed query. Users can then move around the dataset,

both in terms of columns and rows in a traditional display of the data. Browsing may support the same ordering for everybody, as is done in most libraries with *static classification* systems. *Dynamic classification* allows for the ordering of data based on individual preferences, such as will be discussed below.

Retrieval can be implemented by beginning at the starting point and then retrieving those $\mathcal{F}$s that have the highest probability of relevance, given the query treated as providing information about a Bayesian prior, with relevance feedback being incorporated when available (Losee, 1988). This is what occurs in probabilistic information retrieval. Information needs and facts may also be treated as vectors, and the facts that are most similar to the expressed information need, as measured by the cosine between the two vectors, are retrieved (Salton and McGill, 1983).

6 Expected Distance between Adjacent Facts

The distance between two adjacent facts may be computed, consistent with the measurement of the information of an event x as $-\log \Pr(x)$. We suggest that the *expected information dissimilarity*, denoted as $E(\mathcal{I}_{\neq})$, is the expected information associated with a feature times an indicator variable showing the presence or absence of a difference between features. For two metadata vectors, this is computed as

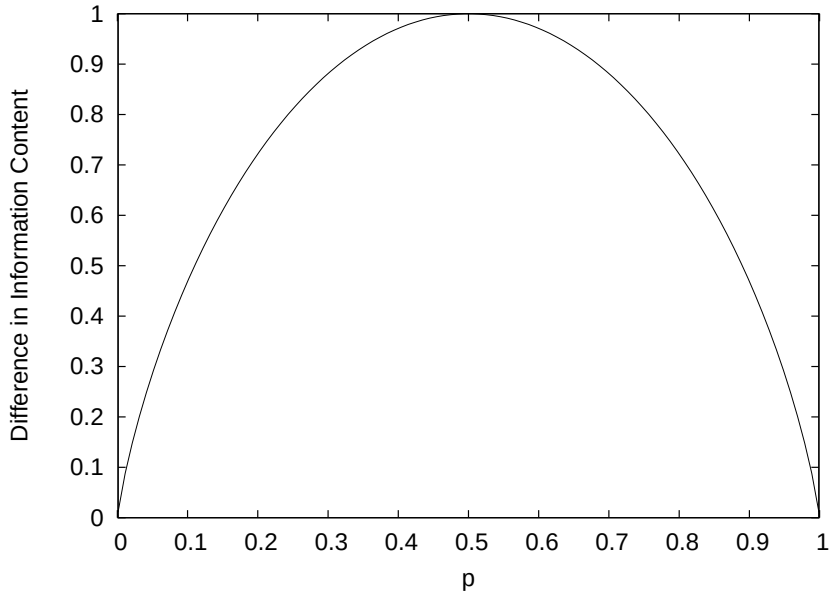
$$E(\mathcal{I}_{\neq}(\mathcal{M}_i, \mathcal{M}_j)) = -\sum_k |m_{i,k} - m_{j,k}| \left(p_k \log p_k + (1-p_k)\log(1-p_k)\right), \quad (1)$$

where $m_{i,k}$ is the value for metadata feature k in metadata vector $\mathcal{M}_i$ and p_k is the probability of feature k occurring. The differences in values for $E(\mathcal{I}_{\neq})$ for a single feature are shown in Figure 1.

Metadata features may be ordered within each column's metadata feature vector so as to help minimize the expected distance between features (Losee, 1992). Features may be ordered randomly, but this leads to suboptimal ordering of facts. Consider a situation where, given a list of foods, the closer a food was to the beginning of the list, the more likely you were to prefer that type of food. Clearly, a random ordering of food will result in the suboptimal (in terms of preferences) serving of food. If, however, the list was ordered by decreasing relative preference of the food, the ordering would be optimal in terms of preference.

We minimize the expected information dissimilarity $E(\mathcal{I}_{\neq})$ between two metadata vectors $\mathcal{M}_i$ and $\mathcal{M}_j$ by placing those individual features in each metadata vector with the lowest expected information dissimilarity furthest to the right in the metadata feature vector, that is, they are placed in the least significant digits positions. Thus, the features with the greatest expected information *similarity* (as changes oc-

Figure 1: $E(\mathcal{I}_{\neq})$, the difference in information content over the range of probabilities for a single feature. Logarithms are computed to base 2.



cur) are changed most frequently, rather than the features whose changes would cause an overall decrease in expected information similarity.

Metadata features on the right (least significant digit) side change most frequently when counting or traversing a list ordered by an enumerating principle, as can be seen informally by counting in the decimal number system and noting how often changes occur in the rightmost column and how often changes occur in the column to its left.

When all the metadata features occur with $p < .5$, $E(\mathcal{I}_{\neq}(m_{i,k}, m_{j,k})$ will be lower for neighboring columns i and j when the rarer terms (with the lower dissimilarity values) are on the right side of the vector and it is most likely that the difference between the neighboring metadata features will be due to a smaller difference on the right rather than a larger difference on the left.

7 Costs of Data Organization

We may optimize feature ordering consistent with economic considerations. A cost, denoted by $\mathcal{C}$, is associated with features having different values in metadata features that are located more than an arbitrarily chosen distance apart. The economic loss of having a 0 for a feature value in the metadata feature $m_{i,k}$ in metadata $\mathcal{M}_i$ and a 1 for

that feature in a $\mathcal{M}_j$ is denoted as $\mathcal{C}_{0,1}$, with a similar cost, $\mathcal{C}_{1,0}$, for the feature k in $\mathcal{M}_i$ having a value of 1 and $\mathcal{M}_j$ having a value of 0.

We suggest that the expected cost of feature k being different in two metadata vectors may be computed as

$$\begin{aligned} E(\mathcal{C}_{\neq}) &= \mathcal{C}_{1,0}p_k(1-p_k) + \mathcal{C}_{0,1}(1-p_k)p_k \\ &= (\mathcal{C}_{1,0} + \mathcal{C}_{0,1})p_k(1-p_k), \end{aligned} \tag{2}$$

the expected cost of dissimilarities associated with feature k. We assume that $\mathcal{C}_{0,0}$ and $\mathcal{C}_{1,1}$ equal 0 and that $\mathcal{C}_{0,1} = \mathcal{C}_{1,0}$.

The cost of placing metadata vectors $\mathcal{M}_i$ and $\mathcal{M}_j$ adjacent to each other, each of length n, is

$$E\big(\mathcal{C}_{\neq}(\mathcal{M}_i, \mathcal{M}_j)\big) = \sum_{k=1}^{n} |m_{i,k} - m_{j,k}|\, \mathcal{C}_k, \tag{3}$$

assuming that features are independent of each other. This represents the sum of costs for metadata features that differ in value between m_i and m_j. For notational simplicity, we have denoted the loss $\mathcal{C}_{1,0}$ as $\mathcal{C}_k$ for feature k.

The relationships between different costs and probabilities are shown in Figure 2.

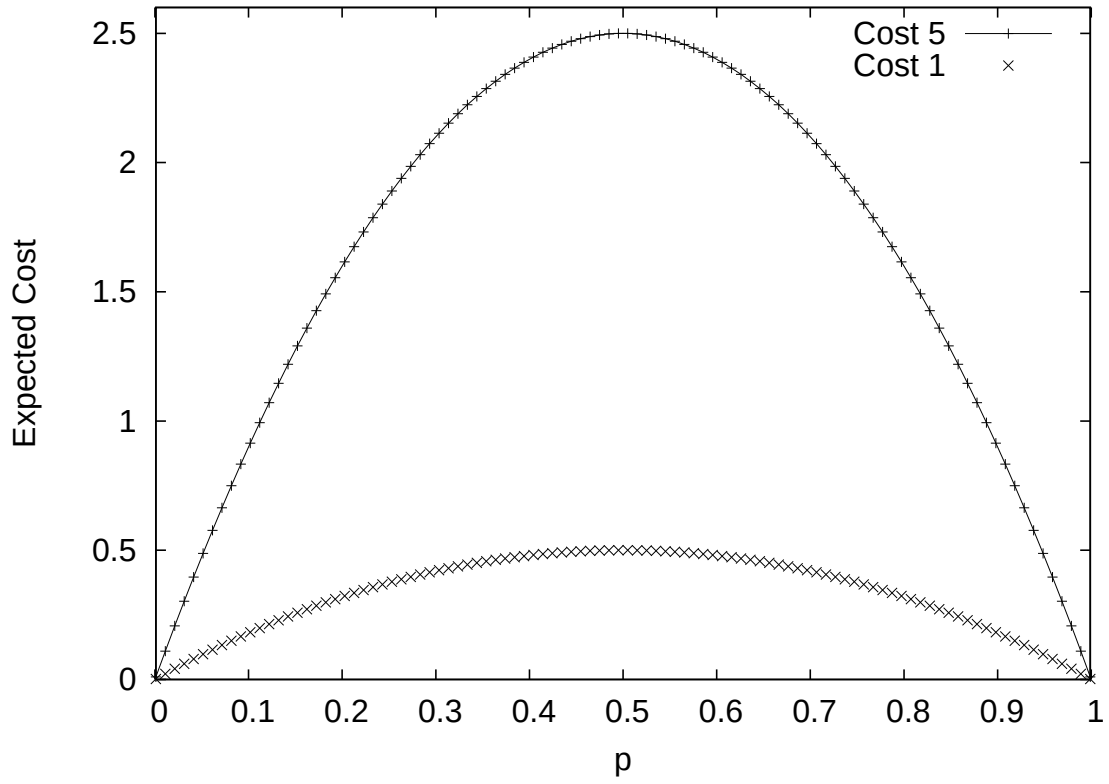
When features were ordered within a metadata vector based on the feature's $E(\mathcal{I}_{\neq})$, the ordering of metadata vectors consistent with the Gray code minimized the expected information dissimilarity between the vectors. If we similarly order features by the expected cost associated with dissimilar metadata features $E(\mathcal{C}_{\neq})$ so that metadata features with the lowest $E(\mathcal{C}_{\neq})$ loss values are placed furthest to the right in the metadata vectors, with features with the highest $E(\mathcal{C}_{\neq})$ loss values are placed furthest to the left, we find that the expected cost of the arrangement has been minimized.

When using the information theoretic expected information dissimilarity, $E(\mathcal{I}_{\neq})$, the probabilities may be easily inferred from the existing data. Costs, however, reflect user values and preferences, and must be obtained directly or inferred from user input to the system. Users might, for example, state that cost for feature i is greater than the cost for feature j. One may also infer user costs by retaining information about user preferences, such as which screen icon was clicked-on first. By averaging these preferences and ordering over time, the preference ordering of the user may be inferred.

8 Measuring Ordering Performance

To study the effectiveness of browsing and retrieval when using ordering, consistent with the Gray code, it is desirable to be able to evaluate the ordering in a repeatable

Figure 2: Expected cost $E(\mathcal{C}_{\neq})$ associated with a one feature document being adjacent to a document with a different feature frequency, with the cost differences being 1 and 5.



and objective manner. One method of evaluating browsing is to study Average Browsing Distance (ABD), the average distance from one relevant item to the next relevant item going in one direction on the browsing path. If we assume that the person is rationally moving in an optimal fashion on the shortest path, the ABD is computed from the length of the browsing path minus the longest gap between relevant items. If we can imagine the 4 letters C, O, K, and E on a single logo on a can or bottle of soda and the distance between the letters, we are ignoring the longest distance, which is probably from the E, around the back of the bottle or can, to the C. This is much longer than the other inter-letter distances between C and O, between O and K, and between K and E. ABD is thus the entire length of the browsing path minus the longest distance between relevant items on the path, divided by one less than the number of items on the path.

One can also examine the dissimilarity between adjacent documents using the techniques for computing expected dissimilarity described earlier. The Average Dissimilarity (AD) represents the average of the dissimilarity between pairs of factual representations around the entire browsing path, with the dissimilarity measured as the Hamming distance. Smaller average dissimilarities indicate that adjacent documents are more alike, providing better browsing performance.

Related to this, we have the Average Information Dissimilarity (AID) which represents the average information difference between adjacent documents, computed over the set of documents. The Average Cost of Dissimilarity (ACD) represents the average economic value associated with the differences between adjacent documents, computed over the set of documents.

9 An Example

Integrating data may be best understood by considering an example showing both structured and non-structured data. The numeric results have been produced from software developed in Mathematica^{TM} which has the ordering capabilities for the probabilistic, information theoretic, and economic models described in the paper. While the data is stored in the Mathematica program, data could be stored in XML format (Blanken et al., 2003) as it is in the author's Nyltiac system (http://Nyltiac.com or http://ils.unc.edu/nyltiac), that allows Gray code based browsing for a variety of uses, such as retrieval, digital libraries, and question answering.

In Table 2 we provide some sample structured data containing two relations. The first has persons' names, their cats' names, and the number of years owner and pet have been together. There is also a second relation containing the cats' names and their favorite foods. Table 3 show the same data placed into the form for factual

Type 1 Record: Person, Cat, Years

Person Name *(person, name, owner)*	*Cat Name* *(cat, name)*	*Years Together* *(yearstogether)*
Lee	Tigger	2
Caitlyn	Sweet Tea	3
Marcia	Sox	1

Type 2 Record: Cat, Favorite Food

Cat Name *(cat, name)*	*Favorite Food* *(favoritefood)*
Tigger	Tuna
Sweet Tea	Chicken
Alice	Chicken

Table 2: Type 1 and 2 Structured records. Below headers are the metadata assigned to the column and below that the data.

representations, as discussed above. The metadata are shown within parentheses. The metadata for the union ($\mathcal{M}_{\mathcal{U}}$) is not shown but is simply the union of $\mathcal{M}_{\mathcal{K}}$ and $\mathcal{M}_{\mathcal{A}}$.

The feature vector being used here contains the following features: owner, author, textblob (binary large object), name, person, cat, yearstogether, favoritefood, and SBN (standard book number). The features author and textblob are used for a following example, and are included here to provide compatibility across examples so that we can merge structured and unstructured data below.

We may add some unstructured data by including the data in Table 4 into the data produced from Table 2.

The browsing performance may be computed using this data and the measures described earlier. The probabilities for features are used to order the features for placement in the feature vector, which becomes (with associated probabilities): name (0.625), cat (0.53125), person (0.4375), owner (0.34375), sbn (0.125), yearstogether (0.09375), favoritefood (0.09375), textblob (0.0625), and author (0.0625).

When ordering features by their probability, the Average Browsing Distance, ABD, is 1 and the Average Dissimilarity, AD, is 1.125 when the variables are placed in de-

Factual Representations

Record	*Record Type*	$\mathcal{M}_{\mathcal{K}}$	*Key*	$\mathcal{M}_{\mathcal{A}}$	*Attribute*
1.	1	(100111000)	Lee-Tigger	(100110000)	Lee
2.	1	(100111000)	Lee-Tigger	(000101000)	Tigger
3.	1	(100111000)	Lee-Tigger	(000000100)	2
4.	1	(100111000)	Caitlyn-SweetTea	(100110000)	Caitlyn
5.	1	(100111000)	Caitlyn-SweetTea	(000101000)	Sweet Tea
6.	1	(100111000)	Caitlyn-SweetTea	(000000100)	3
7.	1	(100111000)	Marcia-Sox	(100110000)	Marcia
8.	1	(100111000)	Marcia-Sox	(000101000)	Sox
9.	1	(100111000)	Marcia-Sox	(000000100)	1
10.	2	(000101000)	Tigger	(000000010)	Tuna
11.	2	(000101000)	Sweet Tea	(000000010)	Chicken
12.	2	(000101000)	Alice	(000000010)	Chicken

Table 3: The feature vector being used here contains the following features: owner, author, textblob (binary large object), name, person, cat, yearstogether, favoritefood, and SBN.

Type 3 Record: SBN, Author, Text

SBN	*Author*	*Text*
(SBN)	*(person,name,author)*	*(textblob)*
1234	Seuss	The cat in the hat
9876	Caitlyn	I love my cats

Factual Representations

Record	*Record Type*	$\mathcal{M}_{\mathcal{K}}$	*Key*	$\mathcal{M}_{\mathcal{A}}$	*Attribute*
13.	3	(000000001)	1234	(010110000)	Seuss
14.	3	(000000001)	1234	(001001000)	The cat in the hat
15.	3	(000000001)	9876	(010110000)	Caitlyn
16.	3	(000000001)	9876	(001001000)	I love my cats

Table 4: The feature vector used in the metadata is: owner, author, textblob (binary large object), name, person, cat, yearstogether, favoritefood, and SBN.

scending order of probability. When the order of features is changed to ascending, the performance drops so that the ABD is 1.07 and the AD is 1.375. In this situation, ordering features in descending order of probability (from left to right) is superior to ordering features in ascending order of probability, supporting the arguments above.

When using the information theoretic model suggested above, features may be ordered consistent with Equation 1 so as to improve performance. The ABD is 1 and the Average Information Dissimilarity, AID, is 0.748 bits when features are ordered consistent with the information theoretic ordering principle suggested above. When the features are placed in ascending order, performance drops and we find an ABD of 1.07 and an AID of 0.988 bits.

Costs were arbitrarily assigned to the features so that *author* was assigned a cost of 10 and all other features were assigned a value of 1. We find that the ABD is 1 when the features are ordered in terms of descending weight and 1.29 when placed in ascending order. The Average Cost of Dissimilarity, ACD, is 2.25 when features are in descending order and an ACD of 2.625 when in ascending order.

In all the cases above, we have found that ordering features based on the principles suggested earlier results in superior organization; placing features in descending order of the appropriate weight produces better browsing.

10 The Adaptive Display of Data

What information should be displayed on a screen? Clearly, the information most likely to be relevant to a user should be displayed, whether the data are facts, images, or movies. By accepting positive or negative user preferences about specific data items, e.g. *Tigger* from Table 2, or preferences about metadata items, e.g., a greater interest in favorite pet foods than in the number of years a pet and owner have been together, systems may adapt their output to present those items most likely to be of interest to an individual user or group of users.

Which material would be preferred by a given user may be inferred from economic feedback, such as that gathered for the purposes above. Using this, specific fields may be seen as being of greater or lesser benefit to the user. If we assume that metadata items are statistically independent of each other, it is reasonable to use economic information about the benefit or cost of each individual metadata item to estimate the relative benefit of an attribute with a specific set of metadata characterizing the attribute. Similarly, changes in databases may provide information theoretic feedback to update collection statistics and feature ordering.

Displays may be viewed as m by n elastic tables. The tables may be generated dynamically, with factual representations being placed appropriately in adjacent rows

or columns. Larger images and textual units may be displayed in a thumbnail form or by using a surrogate, with a "click" on the thumbnail or surrogate producing the full object. The table may be resized based on expressed user preferences, as well as the system's estimate of those preferences.

The system needs to choose either the best n columns for display or choose the single best column as the center column, with those columns placed next to the center column being those that are the successors and predecessors (in the Gray code enumeration) of the best column, or some combination of these two methods. Columns chosen for display may be those with metadata having the highest economic benefit, or those that have the highest probability of being used by the individual or by members of a group to which the individual belongs (an organization). For example, the ordering

$$A \quad B \quad | \quad C \quad D \quad E \quad F \quad | \quad G \quad H,$$

with the displayed columns shown between the vertical bars, might be changed to

$$A \quad E \quad | \quad B \quad C \quad D \quad F \quad | \quad G \quad H,$$

when *E* is assigned a different location in the second, preference-based ordering because of user feedback, effectively changing what is shown in the display window as E is removed (from this display), with *D* and *F* becoming adjacent and *E* shifting to the left between *A* and *B*. By arranging columns consistent with the Gray code based ordering, related information may be displayed, both information specifically requested by users as well as information related to a request, as determined by Gray code ordering (Losee, 2003).

Rows chosen for display may be those that the user has indicated to be of relevance because at this point, or in the past, the user has indicated that specific attributes, e.g., *Bob*, or tuples with specific metadata, e.g., *person*, are of interest. Several factual representations may be joined and placed onto a single line for display; those items whose placement in the table is consistent with the Gray code ordering should be inserted into the table.

Through the ordering of rows and columns for display, a table may be generated from those rows and columns with the highest usability values, along with other related columns. Further research will be needed to determine the exact nature of user preferences for adaptive displays. While the focus of this work is on what we have referred to as an elastic table, users may prefer to have a single fact from a table displayed, joining other related facts from other tables for display, rather than those facts that are most similar.

11 Summary and Conclusions

Placing factual representations into an order consistent with the Gray code and model-determined feature ordering enables us to organize data for browsing. This order decreases the amount of browsing necessary to examine a given amount of information from that consistent with random feature ordering. By using factual representations, structured, semi-structured, and unstructured data can be browsed or displayed together. This technique also allows for the integration of various kinds of solely structured data or solely unstructured data, enabling a number of different tuples to be integrated into a single conceptual linear ordering of all tuples.

Features used in a system which orders factual representations consistent with the Gray code may be placed into different orders. Different criteria for ordering features have been suggested, including probabilistic, information theoretic, and economic. Empirical evidence has been derived from the ordering of the sample data provided in Tables 2 and 3. We were able to show that, when using this data, and for the economic and information theoretic models, placing features in descending order of their weights, as suggested by the arguments earlier, produces the best Average Browsing Distance, ABD, the Average Dissimilarity, AD, the Average Information Dissimilarity, AID, and the Average Cost of Dissimilarity, ACD, values.

References

Arenas, M. and Libkin, L. (2003). An information-theoretic approach to normal forms for relational and XML data. In *Principles of Database Systems*, pages 15–26, New York. ACM Press.

Blanken, H., Grabs, T., Schek, H.-J., Schenkel, R., and Weikum, G. (2003). *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*. Springer, Berlin.

Borkar, V., Deshmukh, K., and Sarawagi, S. (2001). Automatic segmentation of text into structured records. In *International Conference on Management of Data and Symposium on Principles of Database Systems: Proceedings of the 2001 International Conference International Conference on Management of Data*, pages 175–186, New York. ACM Press.

Cover, J. F. and Walsh, B. C. (1988). Online text retrieval via browsing. *Information Processing and Management*, 24(1):31–37.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley Interscience, New York.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classifi cation*. Wiley, New York, 2nd edition.

Faloutsos, C. (1988). Gray codes for partial match and range queries. *IEEE Transactions on Software Engineering*, 14(10):1381–1393.

Gaede, V. and Gunther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231.

Greenberg, J. (2002). Metadata and the world wide web. In Kent, A., editor, *Encyclopedia of Library and Information Science*, volume 72. Marcel Dekker, New York.

Hamming, R. (1986). *Coding and Information Theory*. Prentice-Hall, Englewood Cliffs, N.J., second edition.

Hearst, M., English, J., Sinha, R., Swearingen, K., and Yee, P. (2002). Finding the flow in web site search. *Communications of the ACM*, 45(9):42–49.

Hull, D. (1994). Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland*, pages 282–291, New York. ACM Press.

Jagadish, H. V. (1990). Linear clustering of objects with multiple attributes. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 332–342, New York. ACM Press.

Kowalski, G. (1998). *Information Retrieval Systems: Theory and Implementation*. Kluwer, Boston.

Lee, T. T. (1987). An information theoretic analysis of relational databases, parts I and II. *IEEE Transactions on Software Engineering*, SE-13(10):1049–1072.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, Berlin. Springer.

Losee, R. M. (1988). Parameter estimation for probabilistic document retrieval models. *Journal of the American Society for Information Science*, 39(1):8–16.

Losee, R. M. (1990). *The Science of Information: Measurement and Applications*. Academic Press, San Diego.

Losee, R. M. (1992). A Gray code based ordering for documents on shelves: Classification for browsing and retrieval. *Journal of the American Society for Information Science*, 43(4):312–322.

Losee, R. M. (1993). The relative shelf location of circulated books: A study of classification, users, and browsing. *Library Resources & Technical Services*, 37(2):197–209.

Losee, R. M. (1997a). Browsing document collections: Automatically organizing digital libraries and hypermedia using the Gray code. *Information Processing and Management*, 33(2):175–192.

Losee, R. M. (1997b). A discipline independent definition of information. *Journal of the American Society for Information Science*, 48(3):254–269.

Losee, R. M. (1998). *Text Retrieval and Filtering: Analytic Models of Performance*. Kluwer, Boston.

Losee, R. M. (2002). Optimal user-centered knowledge organization and classification systems: Using non-reflected Gray codes. *Journal of Digital Information*, 2(3). http://jodi.ecs.soton.ac.uk/Articles/v02/i03/Losee.

Losee, R. M. (2003). Adaptive organization of tabular data for display. *Journal of Digital Information*, 4(1). http://jodi.ecs.soton.ac.uk/Articles/v04/i01/Losee.

Manning, C. D. and Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Mass.

McCray, A. T., Gallagher, M. E., and Flannick, M. A. (1999). Extending the role of metadata in a digital library system. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries*, pages 190–199, Baltimore, Maryland. IEEE.

Moens, M.-F. (2000). *Automatic Indexing and Abstracting of Document Texts*. Kluwer, Boston.

Morse, P. M. (1970). *On Browsing: the Use of Search Theory in the Search for Information*. MIT Press, Cambridge, Mass.

Sabin, R. E. and Yap, T. K. (1998). Integrating information retrieval techniques with traditional DB methods in a web-based database browser. In *Proceedings of the 1998 ACM Symposium on Applied Computing*, pages 760–766, New York. ACM Press.

Salminen, A., Tague-Sutcliffe, J., and McClellan, C. (1995). From text to hypertext by indexing. *ACM Transactions on Information Systems*, 13(1):69–99.

Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Vasanthakumar, S. R., Callan, J. P., and Croft, W. B. (1996). Integrating INQUERY with an RDBMS to support text retrieval. *IEEE Data Engineering Bulletin*, 19(1):24–33.

Yan, T. W. and Annevelink, J. (1994). Integrating a structured-text retrieval system with an object-oriented database system. In Bocca, J. B., Jarke, M., and Zaniolo, C., editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 740–749, San Francisco. Morgan Kaufmann.