

Rapid Identification of Column Heterogeneity

Bing Tian Dai
National Univ. of Singapore
daibingt@comp.nus.edu.sg

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

Beng Chin Ooi
National Univ. of Singapore
ooibc@comp.nus.edu.sg

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Suresh Venkatasubramanian
AT&T Labs–Research
suresh@research.att.com

Abstract

Data quality is a serious concern in every data management application, and a variety of quality measures have been proposed, e.g., accuracy, freshness and completeness, to capture common sources of data quality degradation. We identify and focus attention on a novel measure, column heterogeneity, that seeks to quantify the data quality problems that can arise when merging data from different sources. We identify desiderata that a column heterogeneity measure should intuitively satisfy, and describe our technique to quantify database column heterogeneity based on using a novel combination of cluster entropy and soft clustering. Finally, we present detailed experimental results, using diverse data sets of different types, to demonstrate that our approach provides a robust mechanism for identifying and quantifying database column heterogeneity.

1. Motivation

Data quality is a serious concern in every data management application, severely degrading common business practices – industry consultants often quantify the adverse impact of poor data quality in the billions of dollars annually. Data quality issues have been studied quite extensively in the literature (e.g., [5, 10, 3]). In particular, a variety of quality measures have been proposed, e.g., accuracy, freshness and completeness, to capture common sources of data quality degradation [13, 17]. Data profiling tools like Bellman [6] compute concise summaries of the values in database columns, to identify various errors introduced by poor database design; these include approximate keys (the presence of null values and defaults in a column may result in the approximation) and approximate functional dependencies in a table (possibly due to inconsistent values). This paper identifies and focuses attention on a novel mea-

sure, *column heterogeneity*, that seeks to quantify the data quality problems that can arise when merging data from different sources.

Textbook database design teaches that it is desirable for a database column to be homogeneous, i.e., all values in a column should be of the same “semantic type”. For example, if a database contains email addresses, social security numbers, phone numbers, machine names and IP addresses, these semantically different types of values should each be represented in separate columns. For example, the column in Figure 1(a) contains only email addresses and is quite homogeneous, even though there appears to be a wide diversity in the actual set of values present. Such homogeneity of database column values has obvious advantages, including simplicity of application-level code that accesses and modifies the database.

In practice, operational databases evolve over time to contain a great deal of “heterogeneity” in database column values. Often, this is a consequence of large scale data integration efforts that seek to preserve the “structure” of the original databases in the integrated database, to avoid having to make extensive changes to legacy application level code. For example, one application might use email addresses as a unique customer identifier, while another might use phone numbers for the same purpose; when their databases are integrated into a common database, it is feasible that the CUSTOMER_ID column contains both email addresses and phone numbers, both represented as strings, as illustrated in Figure 1(b). A third independently developed application that used, say, social security numbers as a customer identifier might then add such values to the CUSTOMER_ID column, when its database is integrated into the common database. As another example, two different inventory applications might maintain machine domain names (e.g., abc.def.com) and IP addresses (e.g., 105.205.105.205) in the same MACHINE_ID column for the equivalent task of identifying machines connected to

CUSTOMER_ID	CUSTOMER_ID	CUSTOMER_ID	CUSTOMER_ID
lkjkjk@321.zzz.info	lkjkjk@321.zzz.info	lkjkjk@321.zzz.info	123-45-6789
h8742@yyy.com	h8742@yyy.com	h8742@yui.com	135-79-2468
kkjj+@haha.org	kkjj+@haha.org	kkjj+@haha.org	159-24-6837
qwerty@keyboard.us	qwerty@keyboard.us	qwerty@keyboard.us	789-12-3456
555-1212@fax.in	555-1212@fax.in	555-1212@fax.in	987-65-4321
alpha@beta.ga	(908)-555.1234	alpha@beta.ga	(908)-555.1234
john.smith@noname.org	973-360-0000	john.smith@noname.org	973-360-0000
jane.doe@1973law.us	360-0007	jane.doe@1973law.us	360-0007
gwb.dc@universe.gov	8005551212	gwb.dc@universe.gov	8005551212
jamesbond.007@action.com	(877)-807-4596	8778074596	(877)-807-4596

(a) (b) (c) (d)

Figure 1. Example homogeneous and heterogeneous columns.

the network. While these examples may appear “natural” since all of these different types of values have the same function, namely, to serve as a customer identifier or a machine identifier, potential data quality problems can arise in databases accessed and modified by legacy applications that are unaware of the heterogeneity of values in the column.

For example, an application that assumes that the CUSTOMER_ID column contains only phone numbers might choose to “normalize” column values by removing all special characters (e.g., ‘-’, ‘.’) from the value, and writing it back into the database. While such a transformation is appropriate for phone numbers, it would clearly mangle the email addresses represented in the column and can severely degrade common business practices. For instance, the unanticipated transformation of email addresses in the CUSTOMER_ID column (e.g., “john.smith@noname.org” to “johnsmith@nonameorg”) may mean that a large number of customers are no longer reachable.

Locating poor quality data in large operational databases is a non-trivial task, especially since the problems may not be due to the data alone, but also due to the interactions between the data and the multitude of applications that access this data (as the previous example illustrates). Identifying heterogeneous database columns becomes important in such a scenario, permitting data quality analysts to then focus on understanding the interactions of applications with data in such columns, rather than having to simultaneously deal with the tens of thousands of columns in today’s complex operational databases. If an analyst determines that a problem exists, remedial actions can include:

- modification of the applications to explicitly check for the type of data (phone numbers, email addresses, etc.) assumed to exist in the table, or
- a horizontal splitting of the table to force homogeneity, along with a simpler modification of the applications accessing this table to access and update the newly created tables instead.

We next identify desiderata that a column heterogeneity measure should intuitively satisfy, followed by a discussion of techniques to quantify column heterogeneity that meet these desiderata.

1.1. Heterogeneity: Desiderata

Consider the example shown in Figure 1. This illustrates many of the issues that need to be considered when coming up with a suitable measure for column heterogeneity.

Number of Semantic Types: Many semantically different types of values (email addresses, phone numbers, social security numbers, circuit identifiers, IP addresses, machine domain names, customer names, etc.) may be represented as strings in a column, with no *a priori* characterization of the possible semantic types present.

Intuitively, the more semantically different types of values there are in a database column, the greater should be its heterogeneity; thus, heterogeneity is better modeled as a numerical value rather than a boolean (yes/no). For example, we can be confident that a column with both email addresses and phone numbers (e.g., Figure 1(b)) is more heterogeneous than one with only email addresses (e.g., Figure 1(a)) or only phone numbers.

Distribution of Semantic Types: The semantically different types of values in a database column may occur with different frequencies.

Intuitively, the relative distribution of the semantically different types of values in a column should impact its heterogeneity. For example, we can be confident that a column with many email addresses and many phone numbers (e.g., Figure 1(b)) is more heterogeneous than a column that has mainly email addresses with just a few outlier phone numbers (e.g., Figure 1(c)).

Distinguishability of Semantic Types: Semantically different types of values may overlap (e.g., social secu-

rity numbers and phone numbers) or be easily distinguished (e.g., email addresses and phone numbers).

Intuitively, with no a priori characterization of the set of possible semantic types present in a column, we cannot always be sure that a column is heterogeneous, and our heterogeneity measure should conservatively reflect this possibility.

The more easily distinguished are the semantically different types of values in a column, the greater should be its heterogeneity. For example, a column with roughly equal numbers of email addresses and phone numbers (e.g., Figure 1(b)) can be said to be more heterogeneous than a column with roughly equal numbers of phone numbers and social security numbers (e.g., Figure 1(d)), due to the greater similarity between the values (and hence the possibility of being of the same unknown semantic type) in the latter case.

1.2. Heterogeneity: Our Solution

Given the desiderata outlined above, we now present a step-wise development of our approach to quantify database column heterogeneity.

A first approach to obtaining a heterogeneity measure is to use a *hard clustering*. By partitioning values in a database column into clusters, we can get a sense of the number of semantically different types of values in the data. However, merely counting the number of clusters does not suffice to quantify heterogeneity. Two additional issues, as outlined above, make the problem challenging: the relative sizes of the clusters and their distinguishability. A few phone numbers in a large collection of email addresses (e.g., Figure 1(c)) may look like a distinct cluster, but should not impact the heterogeneity of the column as much as having a significant number of phone numbers with the same collection of email addresses (e.g., Figure 1(b)). Again, a social security number (see the first few values in Figure 1(d)) may look similar to a phone number, and we would like the heterogeneity measure to reflect this overlap of sets of values, as well as be able to capture the idea that certain data might yield clusters that are close to each other, and other data might yield clusters that are far apart.

To take into account the relative sizes of the multiple clusters, *cluster entropy* is a better measure for quantifying heterogeneity of data in a database column than merely counting the number of clusters. Cluster entropy is computed by assigning a “probability” to each cluster equal to the fraction of the data values it contains, and computing the entropy of the resulting distribution [4]. Consider a hard clustering $T = t_1, t_2, \dots, t_k$ of a set of n values X , where cluster t_i has n_i values, and denote $p_i = n_i/n$. Then the *cluster entropy* of the hard clustering T is the entropy of the cluster size distribution, defined as $-\sum_i p_i \ln(p_i)$. By using cluster entropy, the mixture of email addresses and

phone numbers in column Figure 1(b) would have a higher value of heterogeneity than the data in Figure 1(c), which consists of mainly email addresses.

The cluster entropy of a hard clustering does not effectively take into account distinguishability of semantic types in a column. For example, given a column with an equal number of phone numbers and social security numbers (e.g., Figure 1(d)), hard clustering could either determine the column to have one cluster (in which case its cluster entropy would be 0, which is the same as that of a column with just phone numbers) or have two equal sized clusters (in which case its cluster entropy would be $\ln(2)$, which is the same as that of a column with equal numbers of phone numbers and email addresses). Intuitively, however, the heterogeneity of such a column should be somewhere in between these two extremes to capture the uncertainty in assigning values to clusters due to the syntactic similarity of values. *Soft clustering* has the potential to address this problem; each data value in soft clustering has the flexibility of assigning a probability distribution for its cluster membership, instead of belonging to a single cluster, as in hard clustering. Heterogeneity can now be computed as the cluster entropy of the soft clustering.

1.3. Our Main Contributions

The first contribution of this paper is a measure of database column heterogeneity, namely,

$$\text{Heterogeneity} = \text{cluster entropy of a soft clustering of the data.}$$

The discussion above justifies this definition, and explains how it captures the properties of a heterogeneity measure. However, this general formulation raises two questions:

- How does one pick the “right” soft clustering? Note this is analogous to asking what the right value of “ k ” is for a hard clustering problem.
- How does one define the cluster entropy of a soft clustering? Note that cluster entropy is conventionally defined only for a hard clustering.

To answer the question about the “right” soft clustering, recall that any clustering algorithm must balance two kinds of costs. The first, expressed by k in a hard clustering problem, and some *compression* measure in general, encodes the complexity of the cluster descriptions - the more compact this is, the better. The second cost is a *quality* measure; how well the given clustering captures the data. All clustering algorithms must trade off between these two costs. Our second contribution in this paper is a principled way of choosing a tradeoff point for soft clustering, based on

rate distortion theory. We find the point of *diminishing returns*, i.e., the point at which the benefits achieved by higher quality stop paying for the penalty in compression. In the language of rate distortion theory, this is the point of unit slope on the (normalized) rate distortion curve.

Our third contribution in this paper addresses the question of cluster entropy of a soft clustering, and proposes the use of *mutual information of a clustering* to represent cluster entropy; this measure, when applied to a hard clustering, yields traditional entropy, and thus is a natural relaxation of entropy to soft clusterings.

Our fourth and final contribution in this paper is a detailed experimental evaluation of these ideas. Using diverse data sets of different types, we demonstrate that our approach of using cluster entropy of a soft clustering provides an effective and efficient mechanism for quantifying column heterogeneity.

The rest of the paper is structured as follows. We first describe related work in Section 2. In Section 3, we present our general framework for quantifying column heterogeneity. This framework is instantiated in Section 4, where we also provide details of our algorithm implementation. Section 5 gives the results of the experimental evaluation of our technique.

2. Related Work

Data quality issues have been studied quite extensively in the literature (see, e.g., [5, 10, 3]). A variety of quality metrics have been proposed, e.g., accuracy, freshness and completeness, to associate with and query along with the data [13, 17]. Mihaila *et al.* [13] associate quality parameters with data, and extend SQL to control data retrieval based on the values of these parameters. Widom [17] proposed the *Trio* model for integrated management of data, accuracy, and lineage, focusing on data model (TDM) and query language (TriQL) issues.

When fields have poor quality data, record linkage techniques using approximate match predicates are fundamental [12]. These techniques return pairs of tuples from the tables, each pair tagged with a score, signifying the degree of similarity between the tuples in the pair according to the specific approximate match predicate. Such approximate join operations have received much research attention in recent years, due to their significance and practical importance (see, e.g., [9, 11]).

Data profiling tools like Bellman [6] collect concise summaries of the values of the database fields. These summaries (set and multiset signatures based on min hash sampling and min hash counts) allow Bellman to determine data quality problems like (i) fields that are only approximate keys (the presence of null values and defaults may result in the approximation), (ii) approximate functional dependencies in a

table (possibly due to inconsistent values), and (iii) approximate joinable keys/foreign keys. However, such profiling tools do not currently help with our heterogeneity problem.

3. Our Approach

3.1. Definitions

A clustering of a set of points is a partition of the set into *clusters*. We can think of this as a probabilistic assignment of points to clusters, where the conditional probability $p(t|x)$ represents the probability that the point x is assigned to cluster t . A partition results from assigning only 0 or 1 to these probabilities. In a *soft clustering*, the probabilities can be arbitrary, with the natural restriction that for any x , $\sum_t p(t|x) = 1$.

The *entropy* of a (discrete) probability distribution P is the sum $H(P) = -\sum_i p_i \log p_i$, where p_i is the probability of the i^{th} element. The entropy of a random variable X is the entropy of the distribution $p(X = x)$, (which we will often write as $p(x)$, where the context is clear). Given a (hard) clustering of n points into clusters $T = t_1, t_2, \dots, t_k$, we say that the *cluster entropy* of T is the quantity $H(T) = -\sum_i p_i \log p_i$, where $p_i = |t_i|/n$.

The *mutual information* between random variables X, Y is the expression $I(X; Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$. In this expression, $p(x,y)$ represents the joint distribution between X, Y , and $p(x), p(y)$ are the marginals. Given a soft clustering of points X into clusters T , we can define the joint distribution $p(t,x) = p(t|x)p(x)$. Typically, $p(x)$ will be the uniform distribution, but this may not always be so.

3.2. A Canonical Soft Clustering

There are numerous methods for performing soft clusterings. Perhaps among the most well known among them is the class of methods known as expectation-maximization, or EM [7]. EM returns a probabilistic assignment of points to clusters. The main problem with this method (and others like it) is that they require the user to fix k , the number of clusters, in advance. One can circumvent this problem by finding the “right” value of k using model checking criteria like AIC [1], BIC [14] and others, but these are all based on assumptions about the distributions the data is drawn from, using maximum likelihood methods to estimate the “most likely” value of k .

A different approach is to use the idea of rate-distortion from information theory [4]. There are two parameters that constrain any clustering method. The first is representation complexity R , or how much compression one can achieve by clustering. The second is the quality Q , or how accurately the clustering reflects the original data. Often, it is

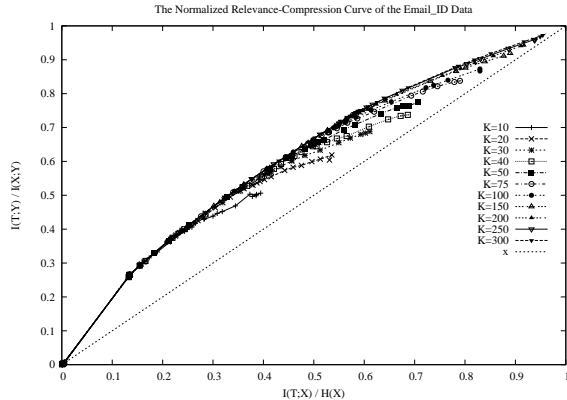


Figure 2. Rate-Distortion curve for a mixture of email addresses and IDs. Note that the tradeoffs achieved are better than those achieved by fixing K , the number of clusters, to any specific value.

more convenient to think of the error E , which is typically some constant minus Q . For any fixed level of compression (this is analogous to fixing k), one can determine the best quality representation, and for any fixed quality level, one can determine the best compression possible.

The rate distortion theorem shows that the optimal representation cost can be measured by the mutual information between the data and the clusters, $R = I(T; X)$. The error E is measured by computing the average distance to the cluster center over all clusters ($E = \sum_{x,t} p(x,t)d(x,t)$). Compression and quality of the optimal solution vary along a concave curve known as the *rate-distortion curve* (see Figure 2). A detailed explanation of rate distortion theory is beyond the scope of this paper; we merely point out that the rate distortion curve can be parameterized by a single Lagrange parameter β , and the points on the curve may be found by optimizing a functional of the form $F = R - \beta \cdot Q$.

Each choice of β corresponds to a point on the rate distortion curve and therefore some soft clustering. Choosing $\beta = 0$ implies that quality does not matter, and thus the optimal clustering is one in which all points are in the same cluster (this corresponds to the origin of the rate-distortion graph, if space is plotted on the x-axis and quality on the y-axis). Letting β go to ∞ is equivalent to making space irrelevant; in this case, all points are placed in separate clusters of their own (this is the top right point of the curve). Note that the slope of the curve at any point is $1/\beta$.

It is important to note that each point on the rate distortion curve is a soft clustering that uses as many clusters as are needed to obtain the optimal value of the rate distortion functional F , for the corresponding value of β . Any fixed choice of the number of clusters to use will ultimately

be suboptimal, as the data separates into more and more clusters. Figure 2 also shows the compression and quality values achieved for clusterings using a fixed number of clusters: for each such number K , there is a point at which the corresponding curve separates from the rate distortion curve, and proceeds along a suboptimal path (less compression, lower quality).

A choice of a clustering is thus made by choosing β . Normalizing the x-axis and y-axis so that the curve goes from $(0, 0)$ to $(1, 1)$, we choose *the point of diminishing returns*; namely the point at which the slope of the curve is one. The reason for our choice is two-fold; the point of diminishing returns, because it has unit slope, is the point after which the benefits of increased quality do not pay for the increasing space needed for the representation. Second, this point is the closest point to the $(0, 1)$ point, which is the point representing perfect quality with no space penalty.

3.3. The Information Bottleneck Method

Our proposed choice of β is quite general, and is independent both of the distance function used, and any distributional assumption. To perform the clustering though, we must fix a representation for the data, and an appropriate distance function. As pointed out by Banerjee *et al.* [2], choosing any Bregman distance yields a meaningful rate-distortion framework, and thus the choice of distance depends on the data representation used.

Choosing ℓ_2^2 as the distance function yields an EM-like method, which would be appropriate for vector data. Our data are strings, which we choose to represent as q -gram distributions. A natural distributional model for string data is a multinomial distribution (for example, the naive Bayes method), and the corresponding Bregman distance is the Kullback-Leibler distance. Using this measure yields a familiar formulation; the information bottleneck method of Tishby, Pereira and Bialek [16]. Some algebra yields a simple description of the rate distortion curve, as the set of points that minimize $\mathcal{F}(\beta) = I(T; X) - \beta I(T; Y)$, where Y is a variable representing the space of all q -grams. Note that the choice of β that yields the point of unit slope on the rate distortion curve is now given by $\beta^* = H(X)/I(X; Y)$. This is a consequence of the fact that $I(T; X)$ is maximized when $T = X$, and thus $I(T; X) = I(X; X) = H(X)$, and $I(T; Y)$ is maximized when $T = X$. We use the iIB algorithm [15] to compute a local optimum of $\mathcal{F}(\beta)$.

3.4. Estimating Heterogeneity

The central idea of this paper is that the entropy of a clustering is a good measure of data heterogeneity. However, cluster entropy is defined only for a hard clustering. Since we know that any hard clustering can be expressed in terms

of probabilistic cluster assignments using only zero and one for probabilities, (and uniform priors on the elements x) we would like the measure we propose to tend towards (hard clustering) cluster entropy in this limiting case.

Using Cluster Marginals The first approach that comes to mind is to determine the marginals $p(t) = \sum_x p(t|x)p(x)$. Our measure is then the entropy of the resulting distribution. Clearly, if all assignments are 0-1 and $p(x) = 1/n$, $p(t) = |\{x|x \in t\}|/n$, and this reduces to $H(T)$. However, by aggregating the individual cluster membership probabilities, we have lost crucial information about the data. Consider two different soft clusterings of two points x_1, x_2 into two clusters t_1, t_2 . In the first clustering, $p(t_1|x_1) = 0.9, p(t_1|x_2) = 0.1$. In the second clustering, $p(t_1|x_1) = p(t_1|x_2) = 0.5$. Both clusterings yield the same marginal $p(t)$ and thus would have the same cluster entropy using the proposed measure. However, it is clear that in the first clustering, x_1 is essentially in t_1 and x_2 is essentially in t_2 (0.9 can be replaced by any number $1 - \epsilon$ for this purpose), and so the cluster entropy should be close to $\log 2$. In the second clustering however, t_1 and t_2 are indistinguishable, which means that there is effectively only one cluster, with a cluster entropy of zero.

Two issues are illuminated by this example. First, aggregating cluster probabilities is not an appropriate equivalent of cluster entropy. Second, *the number of clusters in a soft clustering is an irrelevant parameter*. This is because two clusters having identical cluster membership probabilities will be collapsed (intuitively because they are indistinguishable from each other). Note that the membership probabilities $p(t|x)$ for a point x do not have to be identical for this to happen.

Using Superpositions of Hard Clusterings The second approach we might take is to view the probabilistic assignments as the convex superposition of different hard clusterings. In this view, the assignments reflect the superposition of different “worlds”, each with its own hard clustering. In this case, our strategy is clear; we assign each point to a cluster using the distribution $p(t|x)$, and compute the cluster entropy of the resulting hard clustering. Doing this repeatedly and taking an aggregate (mean or median) gives us an estimate. Note that this approach will yield $H(T)$, as desired, when applied to a single hard clustering, because each point is always assigned to a specific cluster.

However, the second clustering in the above example outlines a problem with this strategy. If a set of points have identical cluster memberships in a set of clusters, then in any particular sample, the points might be distributed among many clusters, rather than all being placed together as they should. For example, in some samples, x_1 might be placed in t_1 , and x_2 might be placed in t_2 , and in others,

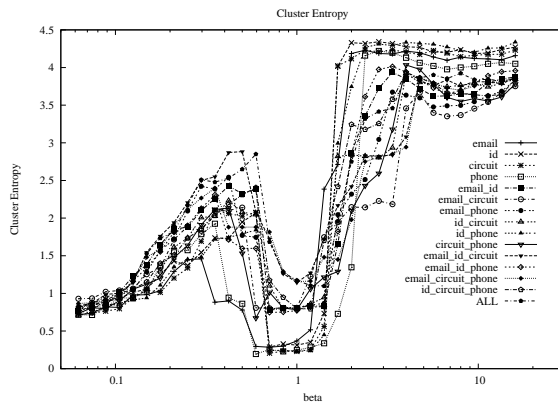


Figure 3. Sampled cluster entropy as a function of β/β^* on a logscale.

they might be placed together. To address this problem, we have to *merge* clusters that have similar cluster membership probabilities, using a threshold parameter and some appropriate merging strategy.

The result of doing this, for multiple data sets and over the range of values for β , is referred to as sampled cluster entropy and is shown in Figure 3. In the figure, the x-axis represents β/β^* .

Observe that for all data sets, this measure exhibits a clear minimum around $\beta = \beta^*$. It turns out that the relative ordering of the heterogeneity values at this value of β correctly reflects the underlying heterogeneity in the data.

The above measure appears quite suitable as a measure of heterogeneity. However, computing it requires at least two parameters; the threshold parameter that decides when clusters are merged, and a parameter deciding the number of samples required to get an accurate estimate. In the absence of a rigorous theoretical analysis, the choice of values for these parameters will inevitably need to be determined experimentally.

Using Mutual Information $I(T; X)$ A more compact solution exploits the relationship between $H(T)$ and $I(T; X)$. Rate-distortion theory tells us that $I(T; X)$ is a measure of space complexity; it is not hard to see that if the assignments $p(t|x)$ are 0-1, then $I(T; X) = H(T)$. Thus, the measure of heterogeneity (and of cluster entropy for a soft clustering) we propose is $I(T; X)$.

A brief illustration of this idea is presented in Figure 4. For each value of β and for multiple data sets, we plot $I(T; X)$ for the soft clusterings obtained using *iIB*. On the x-axis, we plot β relative to the chosen value $\beta^* = H(X)/I(X; Y)$. As β increases from zero, there is a sharp increase in $I(T; X)$ as we approach β^* . This increase occurs for all the data sets, and in roughly the same place.

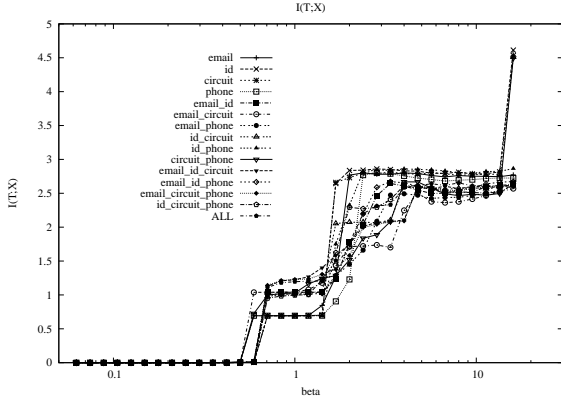


Figure 4. $I(T; X)$ as a function of β/β^* .

The measure of heterogeneity of a data set is then the value of $I(T; X)$ at the point $\beta = \beta^*$ (i.e at $\beta/\beta^* = 1$ on the graph). As we shall see experimentally in Section 5, this measure of heterogeneity correctly predicts the true underlying heterogeneity in the data.

$I(T; X)$ increases monotonically; sampled cluster entropy as described above exhibits a series of (increasing) local minima. It is interesting, and an indication of the robustness of our proposed choice of β , that both measures, while approaching the problem of heterogeneity differently, display the same properties: namely, a sharp change close to $\beta = \beta^*$, and a correct ordering of data sets with respect to their true underlying heterogeneity at this point.

We can now summarize the entire algorithm in Algorithm 1.

Algorithm 1 Computing Heterogeneity.

- 1: Convert input strings to distribution of q -grams. Prepare the data for processing.
 - 2: Compute $\beta^* = H(X)/I(X; Y)$
 - 3: Compute soft clustering using i IB with β set to β^*
 - 4: Estimate heterogeneity by computing $I(T; X)$.
-

Given our choice of $I(T; X)$ as the preferred measure of heterogeneity and of cluster entropy of a soft clustering, when we refer to cluster entropy in the rest of this paper we mean $I(T; X)$, unless otherwise indicated.

4. Methodology

The previous section described our general framework for quantifying column heterogeneity. We now present an instantiation of our framework based on the representation of string data using a multinomial distribution of q -grams, and the details of our algorithm implementation. The input to the algorithm consists of a column of data, viewed as a

set of strings X .

4.1. Preparing The Data

Weighted q -gram Vectors We first construct q -grams for all strings in X . In particular, we construct 1- and 2-grams for all strings. Let the set of q -grams be Y . For each q -gram y , let $f(x, y)$ be the number of occurrences of y in x , and let $p(y)$ be the fraction of strings containing y . We construct a matrix S whose rows are the strings of X and whose columns are q -grams, and the entry $m_{xy} = f(x, y) * w(y) / Z$, where Z is a normalizing constant so that the sum of all entries in M is 1, and $w(y) = H(p(y)) = -p(y) \ln p(y) - (1 - p(y)) \ln(1 - p(y))$. Note that setting $w(y) = -\ln p(y)$ would yield the standard IDF weighting scheme.

Note that standard IDF weighting is not appropriate for our application. IDF weighting was designed to aid in document retrieval, and captures the idea of “specificity”, that a few key phrases could be very useful at identifying relevant documents. IDF weighting captures this by overweighting terms with low relative occurrence.

However, for heterogeneity testing, specificity means that certain rare terms occur only in a few strings. We are not concerned with such outliers, and IDF weighting will only give such strings artificially high importance. Rather, robust heterogeneity testing requires us to identify terms that distinguish large sets of data from each other; the entropy weighting scheme captures this as it is maximized when $p = 0.5$, and decays symmetrically in either direction.

Adding Background Context Any clustering makes sense within a context; a high concentration of points in a small range is significant only if viewed against a relatively sparse, larger background. For example, the collection of strings in Figure 1(a) form a cluster only with respect to the set of all strings. If the background for this data were only the set of email addresses, then this set has no apparent unusual properties.

For heterogeneity testing, an appropriate background is the space of all strings. This needs to be introduced into each data set in order to define the “bounding volume” of the space. As we represent data as distributions, the background consists of random distributions, chosen from the space of all distributions. These are added to the data before soft clustering is performed, and are then removed¹.

It is well known that the uniform distribution over the d -dimensional simplex is a Dirichlet distribution, and thus a uniform sample from this space is obtained by the following simple procedure. Sample d points x_1, \dots, x_d from

¹This is reminiscent of the subtractive dithering used in signal processing to improve quantization.

an exponential distribution with parameter 1, and normalize the values by dividing each by $\sum_{i=1}^d x_i$. The resulting d -vector is a uniform sample from the simplex [8]. A uniform sample from an exponential distribution is computed by sampling r uniformly in $[0 \dots 1]$ and returning the value $\ln(1/r)$.

To generate the background, we use a set of q -grams disjoint from the q -grams in Y , of the same cardinality as Y . Using the above procedure, we generate $|X|$ points, yielding a matrix N that is then normalized so all entries sum to 1. Both S and N have dimension $|X| \times |Y|$.

We fix a parameter $0 < \lambda < 1$ (the *mixing ratio*) that controls the mixture of data and background context. The final joint density matrix M is of dimension $2|X| \times 2|Y|$, containing λS as its first $|X|$ rows and $|Y|$ columns and $(1 - \lambda)N$ as its last $|X|$ rows and $|Y|$ columns. Note that M is a valid joint distribution since its entries sum to 1. We will abuse notation and refer to the rows of M as X and the columns as Y in what follows.

4.2. Computing β^* And Clustering the Data

In Section 3 we derived an expression for the value of β corresponding to the point on the rate-distortion curve that balanced error and compression. This value of β is given by $\beta^* = H(X)/I(X; Y)$. We compute this from the matrix M , using only the data rows and columns. We use the standard empirical estimator for entropy (which treats the normalized counts as fractions).

Given M and β^* , we now run the `iIB` algorithm [15] for computing the information bottleneck. This algorithm is a generalization of the standard expectation-maximization method. Although the algorithm generates a soft clustering, it requires as input a target set of clusters (not all of which may be used in the output). We specify a very large number of clusters ($|X|/2$). Empirically, we see that this is sufficient to find a point on the rate distortion curve. We emphasize here that we *do not* need to fix a number of clusters in advance; the number of clusters that we supply to `iIB` is merely an artifact of the implementation and need only be a very loose upper bound. It only affects the running time of the algorithm, and not the final heterogeneity measure computed.

The output of this algorithm is a soft clustering T , specified as the conditional probabilities $p(t|x)$, from which the cluster masses $p(t)$ and the cluster centers $p(y|t)$ can be derived using Bayes' Theorem and the conditional independence of T and Y given X .

$$p(t) = \sum_x p(t|x)p(x)$$

$$p(y|t) = \sum_x \frac{p(y|x)p(t|x)p(x)}{p(t)}$$

We depict the algorithm in Figure 5, paralleling the steps of algorithm 1.

4.3. Performance Optimizations

Sampling Of Data Columns in a database have many thousands of entries. We sample a small fraction of the entries and compute our heterogeneity estimate on this sample. Since our goal is to detect gross structure in the data, sampling allows us to discard small outlier sets without significantly affecting the results.

Number of Iterations of `iIB` The `iIB` algorithm can often require many iterations to converge. However, experiments show that it almost always converges to within 1% of the final answer within 20 iterations. Hence, we terminate the algorithm after 20 iterations. We present these experiments in Section 5.

5. Experiments

We now present a detailed experimental evaluation of our heterogeneity detection scheme. We will do this using diverse data sets of different types, mixed together in various forms to provide different levels of heterogeneity. We start with a description of the platform and the data sets.

5.1. Platform and Data Sets

The machine we run our experiments on has a Intel(R) Xeon(TM) 3.2 Ghz CPU and 8GB of RAM. It runs Red Hat Linux with the 2.4 kernel and gcc v3.2.3.

We consider mixtures of four different data sets. `email` is a set of 509 email addresses collected from attendees at the 2001 SIGMOD/PODS conference. `ID` is a set of 609 employee identifiers, `phone` is a diverse collection of 3064 telephone numbers, and `circuit` is a set of 1778 network circuit identifiers. Strings in `ID` and `phone` are numeric (`phone` data contains the period as well). Strings in `email` and `circuit` are alphanumeric, and may contain special characters like '@' and '-'.

Each input for our experiments consists of a fixed number of sampled data elements, mixed with the same number of background elements (constructed using the procedure described in Section 4). Elements were sampled from the data sets uniformly, and data mixtures were constructed using equal amounts from each source. A two-set mixture contained equal numbers of elements of each type, and so on. We tried all k -way mixtures for $k = 1, 2, 3, 4$.

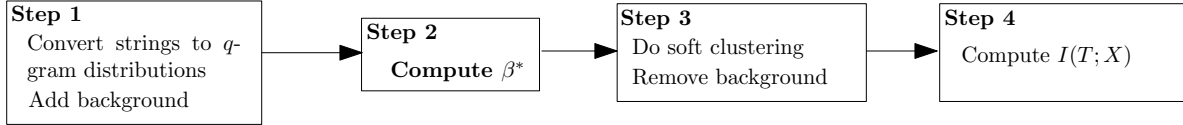


Figure 5. A flowchart for heterogeneity estimation.

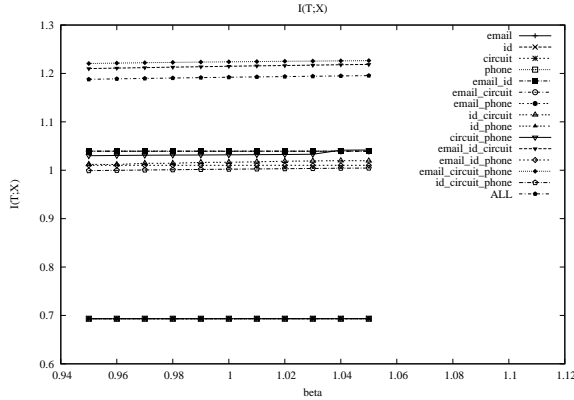


Figure 6. Cluster entropy as a measure of heterogeneity. The x-axis plots β/β^* .

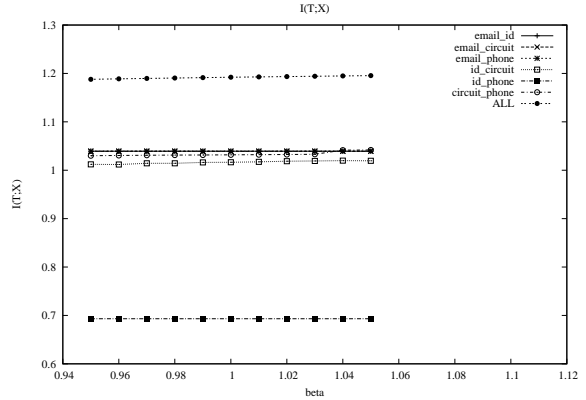


Figure 7. Cluster entropy for mixtures of two sets, and a mixture of all the data sets.

5.2. Validation of Heterogeneity Measure

We start by demonstrating the value of cluster entropy $I(T; X)$ as a measure of heterogeneity. Figure 6 plots the estimated cluster entropy as a function of β/β^* , for values of β around $\beta^* = H(X)/I(X; Y)$. Note that β^* might be different for different data sets. Observe that all the individual data sets have very small cluster entropies, and are well separated from the mixtures. Further, mixtures of two data sets in general have lower cluster entropy than mixtures of three and four. We observe that as the number of elements in the mixture increases, the heterogeneity gap decreases, and that the separations are not strict for the more heterogeneous sets; this is natural, as individual data sets may have characteristics that are similar (e.g., ID and phone).

A closer look at the plot illustrates the idea of well-separatedness and how it is captured in our measure. Consider Figure 7, which plots cluster entropy $I(T; X)$ only for 2-mixtures, and a mixture of all the data sets.

We see a clear separation between the cluster entropy of pairs and the cluster entropy of the 4-way mixture. However, we also notice one 2-mixture that has much lower cluster entropy, much closer to what we would expect for a single data source. The two data sets are ID and phone, both data sets that are predominantly numeric strings, and which often have the exact same number of digits. It is difficult to tell these kinds of strings apart, and as a consequence

their mixture is not viewed as particularly heterogeneous.

The various mixtures whose cluster entropies are depicted in Figures 6 and 7 use equal amounts of data from each source. In Figure 8, we plot cluster entropy for data obtained from a `login` column of a real data set, containing an unknown mixture of values. The heterogeneity of this column was determined to be between a 1-mixture and a 2-mixture. A subsequent inspection of the data revealed a large number of `email` values mixed with a smaller number of ID values.

5.3. Validation of Soft Clustering

Cluster entropy appears to capture our intuitive notion of heterogeneity. However, it is derived from a soft clustering returned by the `iIB` algorithm. Does that soft clustering actually reflect natural groupings in the data? It turns out that this is indeed the case. In Figure 9 we display bitmaps that visualize the soft clusterings obtained for different mixtures. In this representation, columns are clusters, rows are data, and darker probabilities are larger. For clarity, we have reordered the rows so that all data elements coming from the same source are together, and we reordered the columns that have similar $p(t|x)$ distributions.

To interpret the figures, recall that each row of a figure represents the cluster membership distribution of a data point. A collection of data points having the same cluster

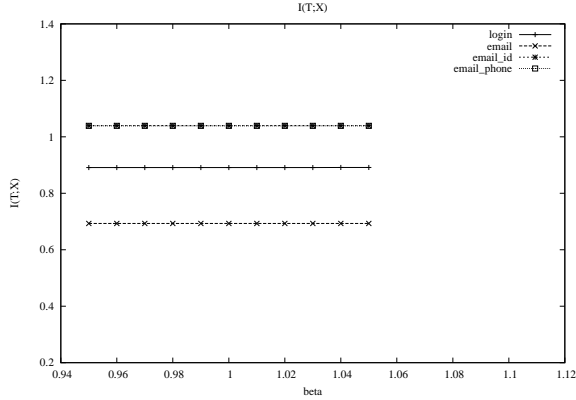


Figure 8. Cluster entropy for an unknown mixture, compared with known mixtures.

membership distributions represent the same cluster. Thus, notice how the clusters separate out quite cleanly, clearly displaying the different data mixtures. Also observe how, without having to specify k , the number of clusters, iIB is able to separate out the groups. Further, if we look at Figure 9, we notice how the clusters corresponding to ID and phone overlap and have similar cluster membership distributions, reinforcing our observation that they form two very close (not well-separated) clusters.

In addition to performing soft clustering and computing cluster entropy, our algorithm adds in background context to aid in the clustering process. In the next experiment, we establish the need for this step.

5.4. Validation of Background Addition

An important aspect of our algorithm is the addition of a background context, as discussed in Section 4.1. We argued that intuitively, the effect of this addition is to “expand” the space being clustered, so a set of points that is highly clustered shows up clearly in contrast to the background. Obviously, if the background level is too high, any data will get swamped, and if too low, will be useless.

Using `email` and `ID`, we illustrate the effect of adding background in Figure 10. Setting the parameter λ to one removes all background from the data. As predicted, when a background context is not added, the data spreads over all clusters in unpredictable ways. Adding it in immediately and predictably collapses the data into a few clusters.

5.5. Sample Size

The default sample size used in our experiments was 200. Increasing the sample size increases the running time of the procedure, but it has the potential for revealing more

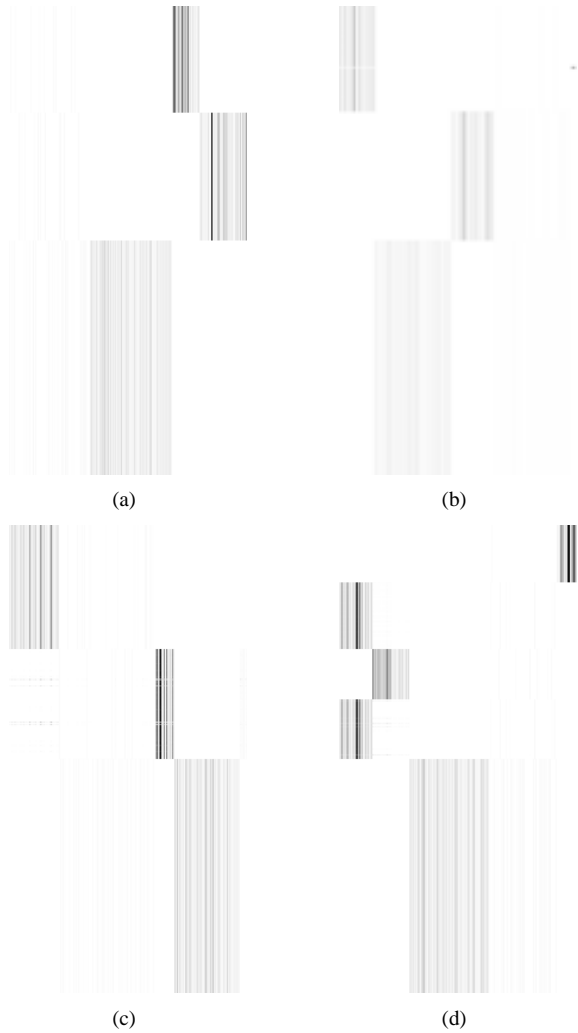


Figure 9. Soft Clustering of `email/ID`, `email/circuit`, `circuit/phone`, `email/ID/circuit/phone` mixtures.

accurate clusterings. 200 samples suffice to capture heterogeneity accurately on our data; in Figure 11, we plot cluster entropy against β for 400-sample mixtures. As always, we use the same number of background elements as data points. Notice that the behavior of the curves is very similar to the behavior with 200 samples (Figure 6).

Conversely, could we have reduced our sample size? The answer, on our data sets, appears to be negative. Figure 12 plots cluster entropy against β for 100-sample data sets. Here we can see that the heterogeneity estimates are starting to deteriorate, most prominently for some 3-way mixtures.

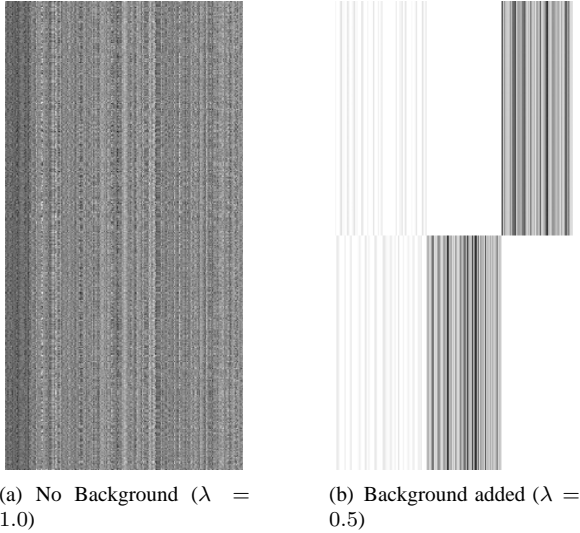


Figure 10. ID clusters with and without background.

5.6. Performance

We now look at the performance of our algorithm. Table 1 breaks down the time taken by various parts of the algorithm. For this experiment, we used an input set of size $2n$, $n = 200, 400$, consisting of n data rows and n background rows, using 1-grams and a maximum of 200 initial clusters to “seed” iIB . Recall that we do not fix the number of clusters finally produced by the algorithm.

We also compare the performance impact of using 2-grams rather than 1-grams when converting the input data to a collection of distributions, in Table 2. The most significant time degradation happens in each iteration of iIB , as the intrinsic dimensionality of the data increases.

As we can see from the above tables, the main bottleneck in the algorithm is an iIB update step. The best way of reducing this time is being able to pick a single value of β to start the iIB iteration, and that is an important feature of our algorithm. Further, each update step of iIB is expensive, and so if we can reduce the number of update steps needed, we can improve the algorithm performance. The number of update steps is controlled by the rate of convergence of iIB . It turns out that in practice, iIB converges rather rapidly, in general requiring 20 iterations or less to converge to within a few percent of its final value. The convergence numbers are shown in Figure 13. Even for far larger instances (not shown in the figure), convergence typically occurs in 30 iterations or less.

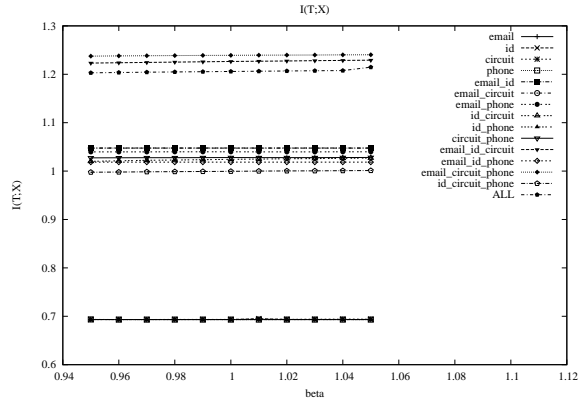


Figure 11. Cluster Entropy with sample size of 400.

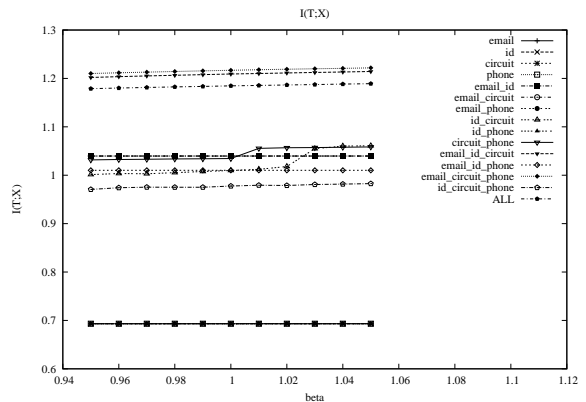


Figure 12. Cluster Entropy with sample size of 100.

6. Conclusion and Future Work

In this paper, we proposed a novel measure of database column heterogeneity as the cluster entropy of a soft clustering of the data, useful to understand data quality issues that can arise when merging data from multiple sources. We addressed the problem of picking the “right” soft clustering by formulating a principled way of choosing a tradeoff point between compression and quality for soft clustering, based on rate distortion theory, as $\beta^* = H(X)/I(X; Y)$. We identified mutual information of a clustering, $I(T; X)$, as a robust measure of cluster entropy of a soft clustering, and demonstrated that it quantifies the column heterogeneity of the underlying data via a detailed experimental study.

There are many further optimization strategies that we might adopt in order to improve the performance and quality of our algorithm. Some that we have experimented with include reducing the alphabet size, trying different term

Mixture	Prepare data		iIB iteration	
	200	400	200	400
email	0.02	0.04	0.42	0.87
ID	0.01	0.01	0.16	0.48
circuit	0.02	0.04	0.23	0.42
phone	0.02	0.03	0.20	0.76
email_ID	0.01	0.03	0.45	1.17
email_circuit	0.02	0.04	0.42	1.24
email_phone	0.02	0.05	0.36	0.94
ID_circuit	0.02	0.03	0.27	0.51
ID_phone	0.01	0.03	0.32	0.72
circuit_phone	0.01	0.02	0.25	0.72
email_ID_circuit	0.01	0.02	0.35	0.75
email_ID_phone	0.01	0.04	0.59	0.70
email_circuit_phone	0.02	0.03	0.36	0.68
ID_circuit_phone	0.01	0.02	0.23	0.45
All	0.01	0.03	0.41	0.67

Table 1. Breakup of running time (in seconds). Generation of background and entropy estimation take negligible time (less than 0.02s) and were omitted.

weighting schemes, and using different models for background context. We plan on exploring this space further, so as to be able to employ our algorithm on databases containing hundreds of thousands of columns.

A central question that comes out of our work is the role of β^* and cluster entropy $I(T; X)$ in determining “good” clusters. Our choice of β^* was based on intuitive arguments about the tradeoff between quality and compression. This was validated by the behavior of $I(T; X)$ for different mixtures of data sets at, and in the vicinity of, β^* . A better understanding of this phenomenon might lead to deeper insights into the way soft clusterings evolve, as well as what structural properties are captured by cluster entropy.

References

- [1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.
- [2] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with bregman divergences. *J. Machine Learning Research*, 6:1705–1749, 2005.
- [3] C. Batini, T. Catarci, and M. Scannapieco. A survey of data quality issues in cooperative information systems. In *ER*, 2004. Pre-conference tutorial.
- [4] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.
- [5] T. Dasu and T. Johnson. *Exploratory data mining and data cleaning*. John Wiley, 2003.

Mixture	Time per iIB iteration	
	1-grams	2-grams
email	0.42	5.55
ID	0.16	1.06
circuit	0.23	1.44
phone	0.20	1.48
email_ID	0.45	5.12
email_circuit	0.42	4.99
email_phone	0.36	5.08
ID_circuit	0.27	1.38
ID_phone	0.32	1.41
circuit_phone	0.25	1.47
email_ID_circuit	0.35	4.58
email_ID_phone	0.59	4.74
email_circuit_phone	0.36	4.63
ID_circuit_phone	0.23	1.47
All	0.41	4.03

Table 2. Time (in seconds) per iIB iteration when using 1-grams and 2-grams.

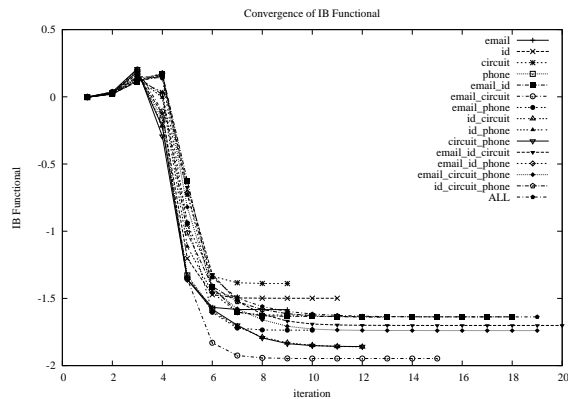


Figure 13. Rate of Convergence of iIB.

- [6] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapyuk. Mining database structure or how to build a data quality browser. In *SIGMOD*, 2002.
- [7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977.
- [8] L. Devroye. Non-uniform random variate generation. <http://cgm.cs.mcgill.ca/~luc/rnbookindex.html>.
- [9] V. Ganti, S. Chaudhuri, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.
- [10] T. Johnson and T. Dasu. Data quality and data cleaning: An overview. In *SIGMOD*, 2003. Tutorial.
- [11] N. Koudas, A. Marathe, and D. Srivastava. Flexible string matching against large databases in practice. In *VLDB*, 2004.
- [12] N. Koudas and D. Srivastava. Approximate joins: concepts and techniques. In *VLDB*, 2005. Tutorial.

- [13] G. Mihaila, L. Raschid, and M.-E. Vidal. Querying “quality of data” metadata. In *Proc. of IEEE META-DATA Conference*, 1999.
- [14] G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
- [15] N. Slonim. *The Information Bottleneck: Theory and Applications*. PhD thesis, The Hebrew University, 2003.
- [16] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.
- [17] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, 2005.