TOWARDS A THEORY OF DATA DISCOVERY:
Category Theory and the DataBridge Experience

A RENCI Technical Report
TR-23-01

Howard Lander and Hao Xu
Renaissance Computing Institute, University of North Carolina at Chapel Hill, NC, USA

Arcot Rajasekar
Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC, USA
School of Information and Library Science, University of North Carolina, Chapel Hill, NC, USA

Corresponding author: Howard Lander, Renaissance Computing Institute, University of North Carolina at Chapel Hill, 100 Europa Drive, Suite 540, Chapel Hill, NC, USA;
Email: howard@renci.org

How to cite this paper: Lander, H., Xu, H., & Rajasekar, A. (2023). *Towards a Theory of Data Discovery: Category Theory and the DataBridge Experience*. Renaissance Computing Institute. https://doi.org/10.7921/YC1S-WX81

renci

www.renci.org

Abstract

As part of the recent "Big Data" revolution, there has been an explosion in the production of scientific datasets. Because of the power of these datasets to assist in additional scientific research and the cost involved in producing these datasets, maximizing their usage is both an economic and scientific imperative. But the sheer number of these datasets has made, in many disciplines, locating data of interest a tedious time-consuming task without guarantee of success. This difficulty has motivated several efforts to ease the task of data discovery. One of the earliest of these efforts was the DataBridge, which uses various techniques to impute relevance amongst datasets. While developing the DataBridge system, we have integrated several seminal concepts that we believe are necessary for discovering data from a diverse digital corpus. In this paper, we discuss our approach, a category-based theoretical justification for that approach and a set of abstract concepts that we believe are central to the data discovery process.

Towards a Theory of Data Discovery: Category Theory and the DataBridge Experience

Data Discovery

Data discovery is a user-driven process of searching for specific items or patterns in single data sets or collections of multiple datasets. In our model, data discovery is finding one or more datasets of interest from an uncomfortably large corpus. This is similar to finding a web page on the World Wide Web using a search engine. The most common WWW searches are keyword-based, with the optional addition of wild cards and regular expressions. An extension of the keyword search is the faceted search, where one may drill down into a particular schema to refine the search for a document of interest. In this paper, we discuss data discovery for scientific datasets. Scientific data is mostly non-textual and is not easily searched directly by keyword or faceted search. Instead, data discovery in scientific datasets is made through searches performed using dataset-associated metadata. Possible metadata sources include direct extraction from the datasets (as is possible for DICOM, FITS, or EXIF files), ingestion of text metadata associated with the file in situ, and generation of the metadata either from observations of user interactions with datasets or from one or more datum in the datasets.

An additional possibility is to exploit the results of domain-specific algorithms used by researchers in analyzing their data to find other data of interest. In this model, the analysis results constitute a dataset signature, which can then serve as metadata to compare datasets to each other. The application of analytical algorithms to generate metadata for aiding search is termed “deep indexing” because it provides a level of discovery based on the data itself that cannot be achieved using textual or schematic metadata. As long as the signatures generated by deep indexing have a partial order, they can be used as an input to similarity algorithms, whose results can be used to define source dataset clusterings. Google’s page ranking for links and restaurant recommendations, based on user star ratings, provides good examples of discovery environments where similarities between entities are based not only on the entities themselves but also on a derived space where comparisons are possible.

While the need for scientific datasets discovery processes is relatively new, data discovery processes have been previously applied to other forms of stored information. These existing data discovery workflows can be separated into four distinct classes:

1. Internet search engines: A search where users look for web pages and documents stored across the World Wide Web using a keyword-based search model.

2. Structured and semi-structured data: A search using relational and XML databases (variations of these, in terms of column store, key-value pairs, and document databases, are also widely used). The data cube or data warehouse model is an extension of this class, where a deeper search occurs on different aggregated facets.

3. Often called search-driven analytics: A search that is becoming more prevalent and is based on user-guided, visual analytics for business intelligence. In this model, data warehouse-type analytics are combined with predictive analytics to provide deeper search for patterns in data.

4. Pattern recognition: A type of search that analyzes data to define abstract patterns. Facial recognition, scene recognition, and regular expression-based identification are examples of this kind of search.

Another form of discovery (though not generally used for data discovery) is based on profiling. In this type of discovery, termed predictive analytics, unsupervised learning is used to classify data into distinctive clusters. The data for this type of discovery is tabular and numerical.

We believe all data discovery workflows can be captured within a single model based on category theory. To define such a model, we consider the DataBridge as an exemplar for these systems and develop a category-theoretic justification of its properties. Different discovery mechanisms can be viewed as subsets of such formalism. The next section introduces the DataBridge (*Big Data: DataBridge*, 2019; Rajasekar, Kum, et al., 2013;

Rajasekar, Sankaran, et al., 2013) system. Section 3 discusses of category theory and our model based on the DataBridge. Section 4 discusses how the DataBridge expresses the theory, and Section 5 concludes with further challenges.

The DataBridge

The DataBridge platform provides data discovery by creating signatures from datasets and using them to compare datasets across a corpus. These signatures can be defined in terms of metadata or created by applying analysis on the data themselves. Importantly, such signatures are morphisms for their datasets, such that similarities at the signature level imply similarities at the data level. Signatures are also comparable. They are comparable to each other, and comparison algorithms can be created to calculate semantic distances between different signatures. Combining these two properties implies that we can analyze a corpus of datasets and represent them by a searchable, undirected, weighted network based on their signatures. In these networks, each node represents a dataset, and the weighting of the edges connecting each pair of nodes is a measure of the similarity of the datasets. Once the network is built, users can query the network for datasets similar to one of interest.

Since its inception, the goal of the DataBridge system has been to build a domain-agnostic platform for data discovery. To achieve this goal, DataBridge was designed with the flexibility to accommodate a variety of user-provided algorithms at each processing stage. This contrasts traditional web search engines, such as Google or Bing, which allow some parameterization of the search, but whose processes and algorithms are somewhat opaque to the user.

Fundamental to the DataBridge is the presumption that each domain community understands its data better than anyone else. To exploit this knowledge, the DataBridge specifies software interfaces for each step in the network-building process. Once the data-specific knowledge is expressed as analytical algorithms (coded in a

programming language such as Java) conforming to the defined interfaces, the platform can execute these algorithms to construct the network. To instantiate this process, we needed to define, at some level of formality, a theoretical understanding of the framework under which data discovery should proceed. The DataBridge is our implementation of this understanding. We believe most, if not all, data discovery systems conform either explicitly or implicitly to some aspects of this framework, even if their implementation lacks the flexibility of the DataBridge.

Document Discovery and Information Retrieval

Several discovery and search mechanisms have been applied to discover unstructured datasets, such as texts, documents, images, and other similar material. We briefly discuss several below, comparing their models and paradigms to the various steps in our discovery theory. Even though the DataBridge system can be used for discovering similarities between documents and texts, it is expressly designed for scientific dataset comparison. This dataset discovery type can be quite complicated, as syntactic-level comparisons may not yield desired results. Moreover, the algorithms used for each dataset type will be different based on the domain and type of data. But, as we discuss below, there are similarities between the steps used in document discovery and the DataBridge discovery framework.

Information retrieval, as noted in (Baeza-Yates, Ribeiro-Neto, et al., 1999), deals with representation, storage, organization of, and access to information items, such as references to real documents, documents themselves, or even single paragraphs, as well as Web pages, spoken documents, images, pictures, music, video, etc. Unlike databases, the representation of the corpus and query are imprecise and ambiguous. IR systems try to provide the best possible matches by estimating their underlying probability of relevance to the submitted query. Relevance is a very important aspect of information retrieval as it provides a way of scoring and ranking the calculated answer set in order of the preferences expressed in the query.

The vector-space paradigm (Salton, Wong, & Yang, 1975) is an information retrieval technique using a weighting scheme that considers (a) local information from individual documents (e.g., term frequency, keyword density, etc.) and (b) global information from a collection of documents (e.g., inverse document frequency, web page reputation, etc.). The weights represent documents, and the queries occur in an n-dimensional space where each term is a dimension in a bag of words/keywords. So a document dj, with n number of terms, can be represented as a point or vector with coordinates dj(w1,j, w2,j ... wn,j). A query's coordinates are defined as q(w1,q, w2,q ... wn,q). Vectors have both direction and magnitude. The relevance of a document, concerning a given query, is given by the cosine similarity between dj and q. Other forms of similarity and normalization (Luo et al., 2018; Sidorov, Gelbukh, Gómez-Adorno, & Pinto, 2014; Singhal et al., 2001), used in information retrieval, are based on algebraic vector-space methods and increasingly on neural networks.

Latent Semantic Indexing (Deerwester, Dumais, Landauer, Furnas, & Beck, 1988; Dumais, 2004) is an important offshoot of the vector-space paradigm. It deals with the two classic problems arising in natural languages: synonymy and polysemy. LSI uses Singular Value Decomposition (SVD) to capture the latent semantic term associations in the relationships between the terms and concepts in a document collection. Using SVD, LSI reduces the term-document matrix of a document collection to a low-rank approximation for a value of k far smaller than the original rank of the term-document matrix. As in any vector-space method, the new k-rank dimensional LSI matrix is used to compute vector similarities. For example, cosine similarities can compute the similarity between a query and a document or between two documents. LSI squeezes the terms-documents matrix down to a smaller k-dimensional space, using the SVD to bring together terms with similar co-occurrences. Comparison in this smaller vector space is faster and produces better retrieval quality. Like LSI, Principle Component Analysis (PCA) (Berry & Martin, 2005) also uses SVD to generate a reduced vector space. LSA

differs from PCA in how the matrix entries are pre-processed before applying the SVD. Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003) is another paradigm that follows similar reduction. Instead of vector space, documents get allocated to a topic space where each document is assigned to a set of topics that through the LDA.

Another type of information retrieval model is based on the probabilistic IR paradigm (Fuhr, 1992; S. Robertson & Zaragoza, 2009; S. E. Robertson & Jones, 1976; Yu & Salton, 1976) in which queries and documents are treated as language models (e.g., query-likelihood model, KL-divergence). The Binary Independence Model (BIM) (S. Robertson & Zaragoza, 2009) and the BestMatch 25 (BM25) are well-known methods based on the probabilistic relevance principle. These methods compute the probability that a document d is relevant to a given query q. The probabilistic IR paradigm also uses the term-document matrix, as in vector-space methods, but computes probabilities (retrieval status values) based on the given query.

The steps used in the vector-space paradigm can be mapped into the morphism steps in the DataBridge discovery theory, discussed in the next section (see Figure 1). The vector-space algorithm is a two-step process: (1) converting and refining documents into a vector space (bag of keywords, TF, TF-IDF, etc.) and (2) normalizing this matrix into a form suitable for comparison, using similarity measures such as cosine similarity. Converting a text document into a vector is like the metadata generation step in the DDM. The vector matrix is a meta representation of the document set. Any comparison made in this vector space is akin to performing a similar comparison on the full document set. The second step, converting a term-document matrix into a cosine similarity or a normalization matrix (or eigenvectors), is an example of signature generation. The LSI paradigm also has similarities to the DataBridge theory. The smaller k-rank dimensional matrix can be viewed as extracted metadata from the document set used for comparison. Similarly, topics identified by LDA are also metadata that provides a clustering for the document set. As mentioned in

(S. Robertson, 2004), IDF can be shown to have a theoretical underpinning (not just a heuristic justification) with regards to the relevance weighting theory (S. E. Robertson & Jones, 1976) and the retrieval status values of the BM25. Robertson (S. Robertson, 2004) concludes, "the theoretical argument behind BM25 can be seen to justify TF*IDF, or in some sense to explain why it works as well as it does, in just the same way that the relevance weighting theory explains IDF on its own." We believe the properties of the morphisms developed in our theory can also justify the soundness of the TF*IDF-based vector-space paradigms. Showing the relevance weighting formulas have the properties of the morphisms, defined below, will be a useful future exercise.

In *Topic Modeling with More Confidence: A Theory and Some Algorithms* (Nguyen, Long, 2015; Tang, Meng, Nguyen, Mei, & Zhang, 2014), authors define a theory for topic modeling, such as LDA, using posterior contraction analysis of topic polytope with a metric to capture the (contracting mass of) neighborhood centered at the true topic values. They use this theory to guide topic modeling for a given corpus and show that properties, such as number of documents, length of documents, and number of topics, are important. Then they conclude that LDA works well when the topics are well-separated, and each document is associated with a few topics. This parametric characterization for LDA might be mappable to the failure or success of the three aspects of the data discovery morphism developed in this paper.

A Formal Model for Data Discovery Systems

The key to DataBridge's discovery process is the similarity analyses at the heart of the system, which uses a representation of the dataset rather than the dataset itself. As described above, this alternate representation, known as a signature, can be derived from the data itself or from data usage, metadata, or other attribute. This signature is the input to the similarity algorithms. We want to study the properties of these signature transformations as a basis for discovery systems. In this section, we define a category theory-based model that applies to our system and the broader set of all

classes of data discovery systems.

Category theory (MacLane, 1971) has been used for modeling systems in multiple scientific disciplines, including physics (Coecke, 2006) and biology (Rosen, 1958). In terms of knowledge representation, category theory has been used to model databases (Spivak, 2009), information visualization (Vickers, Faith, & Rossiter, 2013) and faceted search (Harris, 2015). Category theory provides a very good tool to model the transformations required by the DataBridge.

Before going into the technical details of our model, we explain the key concepts. The model is based on categorically enriched directed graphs, which are directed graphs where the edges between two vertices form an "edge object", which, similar to a hom object, is an object of a category. We use categorically enriched directed graphs to model two domain types: (1) the *data domain*, denoted by c– a domain that is the set of original datasets and their relations, where each vertex is a dataset, and each edge object is a relation from one dataset to another; and (2) the *search domain*, denoted by m. In the case of DataBridge, m comprises signatures and relations among the signatures. Each vertex is a signature, and each edge object is a relation from one signature to another. The edge objects themselves are objects of a category. The category represents relations between edge sets. One of the goals of transforming datasets into signatures is to simplify the computation of similarity while preserving certain aspects of the relations between the relations from datasets to datasets. We choose those transformations that reflect the relations between edge objects. This way, when we run similarity algorithms based on the relations between edge objects in the search domain, any relation we find can be reflected back to the data domain.

There are three aspects to be modeled:

1. Simplification: We only need to preserve the aspects of data relevant to our similarity algorithm, discarding all others in our transformation. Generally, the

data domain c is the collection of complete datasets and their complete relations. The discovery domain m is a view of c that captures an aspect or aspects of c. This leads us to define collections of datasets as graphs and use graph homomorphisms.

2. The preservation of certain aspects of relations between relations: For example, in the case of computing the nearest neighbors, the ordering relation between the relations in the data domain must be preserved by the transformation to the ordering relation between relations in the discovery domain. This leads us to define relations of datasets as categorically enriched graphs and use functors from the data domain category to the discovery domain category to model such transformations.

3. Reflecting the relations between relations in the discovery domain back to the data domain. This leads us to use a construct based on a functor that has an "inverse" to model such transformations. This inverse relationship is important to make sure computed similarities in the discovery domain are a true reflection of the data domain concerning that aspect relevant to our signature algorithm.

The combination of these three aspects results in the "functors reflecting composability" construct, introduced below. This construct allows us to define a category of categorically enriched graphs. This category will be used to model some properties of a signature-based discovery system.

We now formalize this idea in the language of category theory. We start with a brief category introduction (MacLane, 1971).

Definition 1 A category $\mathcal{C}$ consists of the following:

1. A collection of objects denoted as $Ob(\mathcal{C})$.

2. For every pair x, y of objects in $Ob(\mathcal{C})$, a collection of morphisms from x to y, denoted as $Hom_{\mathcal{C}}(x, y)$. The morphism can be written as $f : x \to y$ where x is the domain, and y is the codomain of f.

3. For every object x in $Ob(\mathcal{C})$, the identity morphism is $id_x : x \to x$.

4. For every x, y, z in $Ob(\mathcal{C})$, and every pair of morphisms $f : x \to y$ and $g : y \to z$, the composition of f and g: $g \circ f : x \to z$. □

Definition 2 A morphism $f : x \to y$ is an isomorphism, when there exists a morphism $g : y \to x$, such that:

1. $g \circ f = id_x$

2. $f \circ g = id_g$ □

Definition 3 A category is discrete, when for every pair x, y of objects, $\mathrm{Hom}(x, y) = \emptyset$. □

Definition 4 A functor F from category $\mathcal{C}$ to category $\mathcal{D}$ consists of the following:

1. $F_{Ob} : Ob(\mathcal{C}) \to Ob(\mathcal{D})$.

2. For every $x, y \in Ob(\mathcal{C})$, $F_{Ar(x,y)} : Hom_{\mathcal{C}}(x, y) \to Hom_{\mathcal{D}}(F_{Ob}(x), F_{Ob}(y))$, such that:

   - For every $x \in Ob(\mathcal{C})$, $F_{Ar(x,x)}(Id_x) = Id_{F_{Ob}(x)}$.
   - For every $x, y, z \in Ob(\mathcal{C})$, $f : x \to y$, and $g : y \to z$, $F_{Ar,x,z}(g \circ f) = F_{Ar(y,z)}(g) \circ F_{Ar(x,y)}(f)$. □

We follow the convention of omitting subscripts Ob and $Ar(x, y)$ when it is clear, from context, which one we are referring to.

Definition 5 We say a functor $F : \mathcal{S} \to \mathcal{D}$ is full when: for every pair x, y of objects in $Ob(\mathcal{C})$, and every morphism $f \in Hom_{\mathcal{D}}(Fx, Fy)$, there is a morphism $g \in Hom_{\mathcal{C}}(x, y)$, such that $Fg = f$. □

Definition 6 We say that a functor $F : \mathcal{S} \to \mathcal{D}$ is faithful when: for every pair x, y of objects in $Ob(\mathcal{C})$, and every morphism $f \in Hom_{\mathcal{C}}(x, y)$, there is exactly one morphism $g \in Hom_{\mathcal{D}}(Fx, Fy)$, such that $Ff = g$. □

Definition 7 A subcategory $\mathcal{S}$ of a category $\mathcal{C}$ is a category, such that:

1. $Ob(\mathcal{S}) \subset Ob(\mathcal{C})$.
2. For every $x, y \in Ob(\mathcal{S})$, $Hom_{\mathcal{C}}(x, y) \subset Hom_{\mathcal{C}}(x, y)$.
3. For every $x \in Ob(\mathcal{S})$, id_x in $\mathcal{S} = id_x$ in $\mathcal{C}$.
4. For every $x, y, z \in Ob(\mathcal{S})$, $f \in Hom_{\mathcal{S}}(x, y)$, $g \in Hom_{\mathcal{S}}(y, z)$, $g \circ f$ in $\mathcal{S} = g \circ f$ in $\mathcal{C}$. □

Definition 8 We say the subcategory $\mathcal{S}$ is wide, when $Ob(\mathcal{S}) = Ob(\mathcal{C})$ □

Definition 9 The image $\mathrm{Im}F$ of a functor $F : \mathcal{S} \to \mathcal{D}$ is a subcategory of $\mathcal{D}$, such that:

1. $Ob(\mathrm{Im}F) = \{Fx | x \in Ob(\mathcal{C})\}$.
2. For every $x, y \in Ob(\mathcal{C})$, $f \in Hom_{\mathcal{C}}(x, y)$, $Ff \in Hom_{\mathrm{Im}F}(x, y)$.
3. For every $x, y, z \in Ob(\mathrm{Im}F)$, $f \in Hom_{\mathrm{Im}F}(x, y)$, and $g \in Hom_{\mathrm{Im}F}(y, z)$, $g \circ f \in Hom_{\mathrm{Im}F}(x, z)$.
4. No other morphisms belong to any $Hom_{\mathrm{Im}F}(x, y)$ for any $x, y \in Ob(\mathrm{Im}F)$. □

Definition 10 An embedding is a faithful functor injective on objects. □

Definition 11 When the image of an embedding is a wide subcategory, we call it a wide embedding. □

Now that we have introduced the basic concepts required, we can define the formal system used to model data discovery systems, including the DataBridge. We model these systems by constructing a class of mathematical objects that form a category. A starting point for this construction is directed graphs.

Definition 12 A directed graph G consists of the following:

1. A collection of vertices as $V(G)$

2. For every pair x, y of vertices in $V(G)$, a set $E_G(x, y)$ of edges. □

One can see a directed graph is similar to a category but simpler.

Similar to enriched categories (Borceux & Stubbe, 2000), an enriched directed graph generalizes a directed graph by replacing edge sets with edge objects taken from an arbitrary category. To represent similarity, we enrich directed graphs to categorically enriched directed graphs, where the weights form a category, as follows:

Definition 13 Given a category $\mathcal{V}$, a directed graph G enriched in $\mathcal{V}$, called a $\mathcal{V}$-graph, consists of the following:

1. A collection of vertices as $V(G)$

2. For every pair x, y of vertices in $V(G)$, an object in $\mathcal{V}$, denoted $E_G(x, y)$. □

For example, a simple graph S can be constructed as an enriched directed graph, where the edge objects are taken from the discrete category $\{\emptyset, \{*\}\}$ with $E_S(x, x) = \{*\}$ for every $x \in V(S)$.

We will construct a category in which the objects are not limited to $\mathcal{V}$-graph for a $\mathcal{V}$, but all small categories. It is insufficient to use $\mathcal{V}$-graph homomorphisms[1] as morphisms. We need a general definition of morphisms from a $\mathcal{V}$-graph to a $\mathcal{W}$-graph for arbitrary $\mathcal{V}$ and $\mathcal{W}$. This concept is used to formalize the key concept in DataBridge. To tie it back to our DataBridge concepts, the category of the edge objects

[1] The most straightforward way to define the category of $\mathcal{V}$-graphs is via enriched directed graph homomorphisms. Enriched directed graphs form categories, where the morphisms are enriched directed graph homomorphisms.

Definition 14 An enriched directed graph homomorphism between $\mathcal{V}$-graphs F and G is a graph homomorphism f, such that:
For every pair x, y of vertices in $V(F)$, a morphism $f_{x,y} : E_F(x, y) \to E_G(f(x), f(y))$ □

of the data domain is generally more complicated than the category of the edge objects of the search domain. We need to formalize the idea of a partial edge objects category view. The idea is to simplify objects in the source category by only looking at some aspects of the objects, similar to "projecting" in linear algebra. Then, we discover relationships among the edge objects in the search domain. We want to make sure these relationships are reflected back to the data domain.

We define the concept of functors reflecting composability. Given a functor $F : \mathcal{C} \to \mathcal{D}$, we want to be able to select a finite set of objects in $\mathcal{C}$, map them to $\mathcal{D}$, discover a subcategory in $\mathcal{D}$ spanning these objects, and reflect that subcategory back to $\mathcal{C}$.

Definition 15 A functor $F : \mathcal{C} \to \mathcal{D}$, reflects composability when for every finite category $\mathcal{E}$ and its wide, discrete subcategory $\mathcal{I}$, with inclusion functor C, every functor $G : \mathcal{I} \to \mathcal{C}$, and every functor $H : \mathcal{E} \to \mathcal{D}$, such that

$$FG = HC \tag{1}$$

there is a functor $F^* : \mathcal{E} \to \mathcal{C}$, such that :

$$F^*C = G \tag{2}$$

$$FF^* = H \tag{3}$$

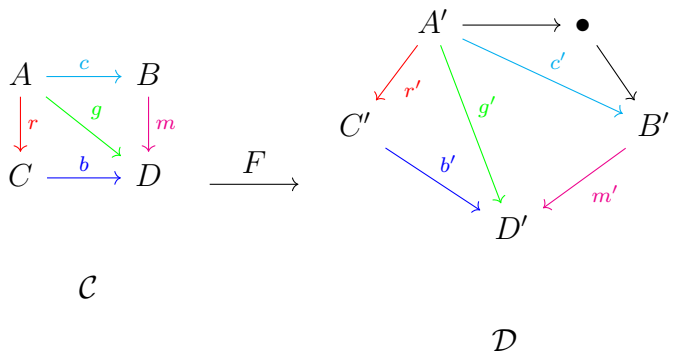
$$\begin{array}{ccc} \mathcal{I} & \xrightarrow{G} & \mathcal{C} \\ {\scriptstyle C}\downarrow & \nearrow_{F^*} & \downarrow{\scriptstyle F} \\ \mathcal{E} & \xrightarrow{H} & \mathcal{D} \end{array}$$

□

We say G is the query, picking a finite bag of objects in $\mathcal{C}$ and H, with the answer returning a set of relations between the objects.

Let's look at some examples, illustrating the functors' roles in the definition.

Example 1 *Functor Reflecting Composability*: Let F be the functor:



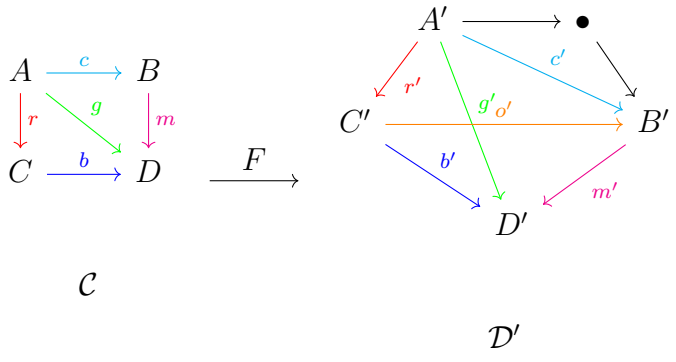
where $F(r) = r'$, $F(g) = g'$, $F(b) = b'$, $F(c) = c'$, and $F(m) = m'$.

F reflects composability: Eq (1) from Definition 15 says, for any object E of $\mathcal{E}$, $H(E)$ must be one of A', B', C', D', so H selects a subcategory of the full subcategory of $\mathcal{D}$, spanned by A', B', C', D', but any subcategory of such a subcategory is isomorphic to a subcategory of $\mathcal{C}$. As a result, we can choose F^*.

Note: the extra black node in $\mathcal{D}$ represents nodes without a corresponding node in $\mathcal{C}$, under the functor F. □

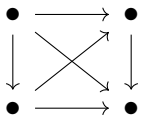
The following is an example of a functor that does not reflect composability.

Example 2 *Functor not Reflecting Composability*: Let F be the functor:



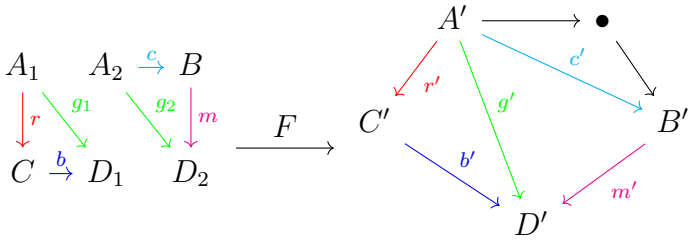
where $F(r) = r'$, $F(g) = g'$, $F(b) = b'$, $F(c) = c'$, and $F(m) = m'$.

F does not reflect composability. Notice the only different between $\mathcal{D}'$ and $\mathcal{D}$, from Example 1, is the addition of a morphism o'. This allows us to choose $\mathcal{E}$ to be the category:

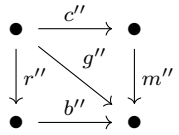


which is not isomorphic to any subcategory of $\mathcal{C}$. Hence, we cannot find any F^*. □

Example 3 *Not all functor-to-full subcategories reflect composability*: Let F be the functor:

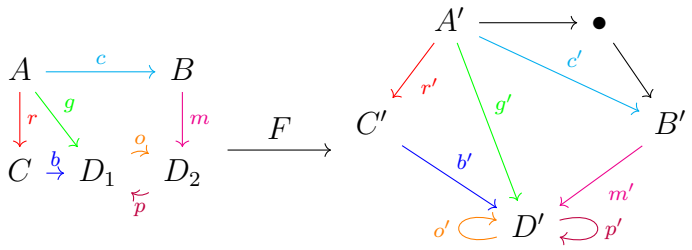


where $F(r) = r'$, $F(g_1) = F(g_2) = g'$, $F(b) = b'$, $F(c) = c'$, and $F(m) = m'$. If we choose $\mathcal{E}$ to be



and H to be $H(r'') = r', H(g'') = g', H(c'') = c', H(b'') = b', H(m'') = m'$, we cannot find any F^*. □

Example 4 *A functor reflecting composability does not have to be injective on objects.* Let F be the functor:



where $F(r) = r'$, $F(g) = g'$, $F(b) = b'$, $F(c) = c'$, $F(m) = m'$, $F(o) = o'$, and $F(p) = p'$. F is a functor reflecting composability, if:

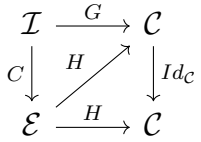
1. o' and p' are left identities of composition, and

2. $o' \circ p'$ and $p' \circ o'$ are identities. □

We now show that the categories and functor that reflect composability form a category.

Lemma 1 *The identity functor reflects composability.* □

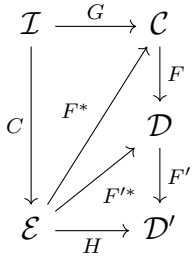
Proof By chasing the following diagram, obtained by setting $\mathcal{D}$ to $\mathcal{C}$, F to $Id_{\mathcal{C}}$, and F^* to H in Definition 15.



■

Lemma 2 *The composition of functors that reflect composability, reflects composability.* □

Proof In the following diagram,



for every G, H, and C satisfying

$$HC = (F'F)G$$

we have by associativity,

$$HC = F'(FG)$$

By the property of reflecting composabilty of F', we can find F'^*, such that

$$F'^*C = GF$$

and

$$F'F'^* = H \tag{4}$$

By the property of reflecting composabilty of F, we can find F^*, such that

$$F^*C = G$$

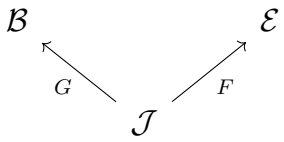
and

$$FF^* = F'^* \tag{5}$$

With these equations, we can prove this lemma. ■

Given the concept of a functor reflecting composability, we can define the concept of a "partial view" of a category, which is partial because it may only concern some, but not all, morphisms in the category. If we consider the hom set between two objects in a category as the total relationship between the objects, the partial view only needs to contain a subset of the hom set. This constraint can be expressed using wide embedding, which allows us to encode a partial view's intuition as subsets on hom sets. On the other hand, we want to be able to work with only partial information from a morphism, as long as it is enough to reflect composability, which may result in two morphisms being mapped to one morphism in the partial view. This is expressed by functor reflecting composability.

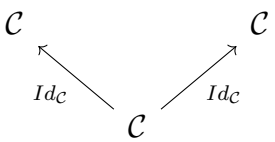
Definition 16 A partial view from $\mathcal{B}$ to $\mathcal{E}$ is a quintuple $(\mathcal{B}, \mathcal{E}, \mathcal{J}, G, F)$,



where $\mathcal{J}$ is a category, G is a wide embedding on objects, and F is a functor reflecting composability. □

We now show the categories and partial views form a category.

Lemma 3 *The following is a partial view:*



□

Proof By Lemma 1 and that the identity functor is a wide embedding from a category to itself. ■

We define the composition of two partial views.

Definition 17 The composition of the partial view

$$(\mathcal{B}, \mathcal{E}, \mathcal{J}, G, F)$$

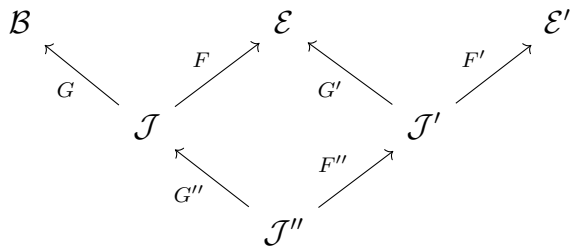
with another partial view

$$(\mathcal{E}, \mathcal{E}', \mathcal{J}', G', F')$$

gives a composite view

$$(\mathcal{B}, \mathcal{E}', \mathcal{J}'', GG'', F'F'')$$

given by



In this common partial view, the three components J'', G'', and F'' are constructed as follows:

Construction of $\mathcal{J}''$.

$$Ob(\mathcal{J}'') = Ob(\mathcal{J})$$

For every $x, y \in Ob(\mathcal{J}'')$,

$$\begin{aligned} hom_{\mathcal{J}''}(x,y) = \{f \in hom_{\mathcal{J}}(x,y) \mid \\ \exists x', y' \in Ob(\mathcal{J}') \exists f' \in hom_{\mathcal{J}'}(x', y') G'(f') = F(f)\} \end{aligned} \tag{6}$$

where the identity and composition follow $\mathcal{J}$.

Lemma 4 *$\mathcal{J}''$ is a category.* □

Proof We only need to prove the existence of identity and composition.

Existence of identity. Since G' is a wide embedding, for every $x \in Ob(\mathcal{J})$, there exists $x' \in Ob(\mathcal{J}')$, such that

$$G'(x') = F(x)$$

Therefore,

$$G'(id_{x'}) = F(id_x)$$

Therefore, $id_x \in hom_{\mathcal{J}''}(x, x)$.

Existence of composition. Given morphisms $f : x \to y$ and $g : y \to z$ in $\mathcal{J}''$, by Equation (6), there must be morphisms $f' : x' \to y'$ and $g' : w' \to z'$ in $\mathcal{J}'$, such that

$$G'(f') = F(f)$$

and

$$G'(g') = F(g)$$

By G''s property of embedding,

$$y' = w'$$

By functoriality,

$$G'(g' \circ f') = F(g \circ f)$$

Laws of identity. Follows from $\mathcal{J}$.

Laws of composition. Follows from $\mathcal{J}$. ■

Construction of G''. G'' is the embedding functor from $\mathcal{J}''$ to $\mathcal{J}$.

Construction of F''. F'' is given by $F''(x) = x'$, such that $FG''(x) = G'(x')$ and $F''(f) = f'$, such that $FG''(f) = G'(f')$.

Lemma 5 *F'' is well defined.* □

Proof We need to prove the existence of x', uniqueness of x', existence of f', uniqueness of f', preservation of identity, and functoriality.

Existence of x'. By G''s property of being surjective on objects, there exists x', such that

$$FG''(x) = G'(x')$$

Uniqueness of x'***.*** By G''s property of being injective on objects, there is a unique x', such that

$$FG''(x) = G'(x')$$

Existence of f'***.*** By Equation (6), for every f in $\mathcal{J}''$, there exists f', such that

$$F(f) = G'(f')$$

By G''''s property of embedding,

$$FG''(f) = G'(f')$$

Uniqueness of f'***.*** Assume that

$$FG''(f) = G'(f'_1)$$

and

$$FG''(f) = G'(f'_2)$$

We have

$$G'(f'_1) = G'(f'_2)$$

By G''s property of embedding,

$$f'_1 = f'_2$$

Preservation of identity. Given $x \in Ob\mathcal{J}''$, by existence there exists f', such that

$$FG''(id_x) = G'(f')$$

Therefore,

$$id_{FG''x} = G'(f')$$

By G''''s property of embedding, we also have

$$id_{FG''x} = G'(id_{x'})$$

for some x'. By uniqueness,

$$f' = id_{x'}$$

Functoriality. Given $f : x \to y$ and $g : y \to z$ in $\mathcal{J}''$, we have

$$\begin{aligned} G'(F''(g) \circ F''(f)) &= G'(F''(g)) \circ G'(F''(f)) \\ &= FG''(g) \circ FG''(f) \\ &= FG''(g \circ f) \\ &= G'(F''(g \circ f)) \end{aligned}$$

By G'''s embedding,

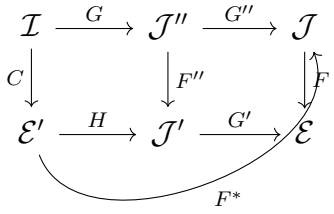
$$F''(g) \circ F''(f) = F''(g \circ f)$$

In fact, this is the pullback of F and G'. □

Lemma 6 *F'' reflects composability.* □

Proof Given a finite, discrete category I, a finite category $\mathcal{E}'$, a functor $G : \mathcal{I} \to \mathcal{J}''$, a functor $H : \mathcal{E}' \to \mathcal{J}'$, and a functor $C : \mathcal{I} \to \mathcal{E}'$ surjective on objects, such that

$$F''G = HC \tag{7}$$



By F's reflecting composability, we have $F^* : \mathcal{E}' \to \mathcal{J}$, such that

$$F^*C = G''G \tag{8}$$

$$FF^* = G'H \tag{9}$$

By Equation (9), for every morphism $f \in Ar(\mathcal{E}')$, we have

$$G'(H(f)) = F(F^*(f))$$

By (6), $F^*(f) \in Ar(\mathcal{J}'')$. Therefore, we can construct a functor $F''^* : \mathcal{E}' \to \mathcal{J}''$, such that

$$F''^*C = G \tag{10}$$

$$FF''^* = H \tag{11}$$

■

Lemma 7 *The composition of partial views is a partial view.* □

Proof By Lemma 6, F'' reflects composability. By Lemma 2, and that F' reflects composability, $F'F''$ reflects composability.
Because the composition of a wide embedding is a wide embedding, GG' is a wide embedding. ■

Having defined "partial views," we can define a category of categorically enriched directed graphs.

As stated earlier, we can model datasets, relations between datasets, and relations between relations by considering a categorically enriched directed graph, where the vertices represent datasets, the edge objects represent relations between them, and the category from which the edge objects are chosen represents relations between relations. Given a categorically enriched directed graph representing the data domain c, we do not want just any morphism entered into any enriched directed graph. A morphism that transforms a graph representing the data domain into a graph representing the search domain must follow certain conditions. We develop information about the relations in the dataset domain using the relations between datasets mapped to the search domain. For this mapping to be valid, it must reflect certain properties. For example, a ranking algorithm in the search domain should be reflected in the data domain. That is, we require morphisms that are *relation-preserving* on some aspects of the edge objects in the opposite direction. These morphisms describe the relationship between the *data domain* and the *search domain*. In other words, we want to rule out the possibility that the *search domain* will contain relations between edge objects that have no basis in the *data domain*. This way, searching in the search domain produces results that reflect back on those in the data domain. The correct morphism d transforms domain c into an enriched directed graph m whose edge objects relations are preserved and reflected between them. That is, the morphism must preserve the relations between datasets as well as reflect the relations of the relations between datasets.

We formalize this as morphisms between categorically enriched directed graphs.

Definition 18 Given categories $\mathcal{A}_c$ and $\mathcal{A}_m$, a *data discovery morphism* (ddm) from an $\mathcal{A}_c$-graph c to an $\mathcal{A}_m$-graph m is a pair (d, P), where d is function from $V(c)$ to $V(m)$, and $P = (\mathcal{A}_c, \mathcal{A}_m, \mathcal{J}, G, F)$ is a partial view from $\mathcal{A}_c$ to $\mathcal{A}_m$, such that for every $a, b \in V(c)$,

$$(F_{Ob}G_{Ob}^{-1})(E_c(a,b)) \quad = \quad E_m(d(a), d(b)) \qquad \square$$

Categorically enriched directed graphs form a category.

Definition 19 DataBridge is defined as follows:

- The objects are enriched directed graphs.
- The morphisms are *data discovery morphisms* (ddm). $\square$

Theorem 1 DataBridge *is a category.* $\square$

Proof Using Lemma 3, we can define an identity ddm for any categorically enriched directed graph. $(id_{V(c)}, id_{\mathcal{A}_c})$, where $id_{V(c)}$ is the identity function on $V(c)$, and $id_{\mathcal{A}_c}$ is the identity partial view on $\mathcal{A}_c$.
Using Lemma 7, we can define a composition of two ddms with matching domains and codomains. Given ddms (d_1, P_1) and (d_2, P_2), the composition of the two ddms is $(d_1 \circ d_2, P_1 \circ P_2)$. Therefore, the categorically enriched directed graphs form a category. ■

Using ddms to build a General Data Discovery Pipeline

Let $d : c \to m$ denote a data discovery morphism defined between the two domains. In the DataBridge, the morphism *ddm* is implemented as a composite of a series of individual morphisms. The original data domain c contains relationships that can be used to explore connections between the datasets. If these relationships are explicit and, therefore, easily discoverable, we don't need to perform any transformations, and the data domain becomes the search domain. In general, however, these relationships are latent, which motivates the need to create the search domain m from the data domain

c. This transformation can require multiple sub-morphisms; these mappings (and any compositions of them) must have certain preservation properties. Next, we look at these mappings in more detail.

Morphism f

The first sub-morphism is the representation of the actual datasets by associated metadata. As noted above, the metadata can come from several different sources. In addition, selected metadata can vary depending on the desired signature and similarity algorithms. This morphism is represented by

$$f : dataset \rightarrow metadata$$

This morphism is generally necessary because the datasets alone are not amenable to similarity comparison. The underlying assumption is that metadata derived by f preserves latent relationships between the original data domain c datasets. For example, consider a corpus of datasets comprising lengthy text documents. In this case, a carefully derived “bag of words” can be regarded as metadata for each document. Comparing the bag of words for similarities between two documents reflects the similarity between the originals.

In some cases, one may even perform further data reduction, restricting the “bag of words” to a “bag of keywords” from a controlled vocabulary and still capturing the similarities between documents. An example is the MEDLINE use of the "Medical Subject Headings" (MeSH) controlled vocabulary, meaning a specific set of terms is used to describe each article in MEDLINE. Familiarity with this vocabulary allows for better, more focused searching.

Moreover, because of the tree structure of the vocabulary, PubMed automatically includes narrower terms in the search if they are available. Inverted indices are another example of a morphism applied to a data store. We can define search and similarities on inverted indices of a corpus of documents instead of making comparisons across the

corpus, obtaining the same results. Database table indexing is a third example of this type of morphism. In this case, searches for values and ranges of values are done on the indexes instead of doing a full table scan.

Morphism g

To calculate similarities between metadata, dataset metadata must be comparable, but the metadata must also be suitable for the desired similarity calculation. Morphism g, with signature

$$g : metadata \rightarrow signature$$

is used to transform the metadata generated by morphism f into a signature conforming to the needs of the desired similarity algorithm. Morphism g is needed, because not all metadata derivations, even if preserving relationships, are easy or accurate on which to perform comparisons. For example, the inverted index is a metadata representation that is not particularly useful for finding similarities. Signature algorithms for categorical data often remove stopwords or perform a stemming operation. More complicated signature algorithms may perform any number of computational steps. For instance, a signature algorithm may produce a signature composed of a vector of high-level concepts, whose absence or presence is determined (for each dataset) using a previously created ontology. Such signature comparisons can be seen in DNA sequence alignments and other edit distance quantifications. Note, in the case of databases, the concept of an index is sufficient, and no signature generation is needed. This is because database searches are based on finding exact matches (which the index can provide), rather than applying a notion of similarity to find similar rows in a database.

Morphism h

The signature morphism g, transforms metadata into an algorithm-ready form. Morphism h, with signature

$$h : signature \rightarrow similarity$$

calculates the similarity values by applying one or more defined similarity algorithms to the generated signatures. Morphism h completes the transformation of the original *data domain* to the *search domain* that is required to represent the latent relationships hidden in the original datasets in an explicit, searchable form. Examples of similarity algorithms include overlap and cosine for categorical and numeric data, respectively, and Hamming distance for vector data.

Figure 1 depicts this entire transformation. That is:

$$d = h \circ g \circ f$$

and captures the mapping

$$d : c \to m$$

Note: the morphisms f, g, and h need not be single functions but can be compositions of functions. For simplicity, we consider each of these as a single function in our discourse below. In section E, we look at the properties necessary for these transformations.

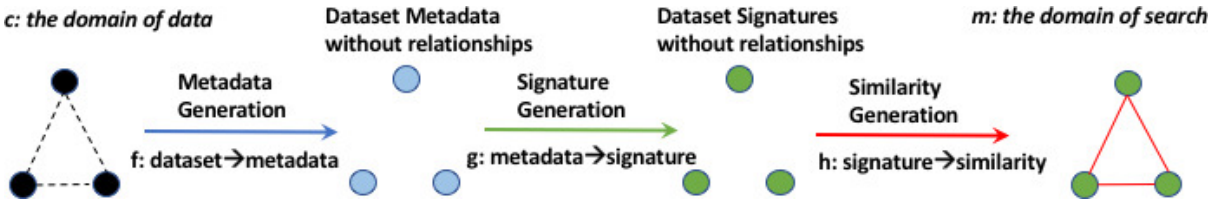


Figure 1. The Data Discovery Morphism $d = h \circ g \circ f$

Applying the Theory: The DataBridge

Basic Operations

With the theoretical background described above, we can describe the basic operations of the DataBridge as follows: First, we instantiate morphism f by extracting or generating appropriate metadata for each dataset in c, the data domain. Using this metadata, we execute an appropriate signature algorithm as morphism g. The signatures generated at this step become the vertices of m, or the search domain. A

similarity algorithm executed on every pair of signatures instantiates morphism h and forms the edges of m. For this m, f can be defined obviously: for each vertex a in c, let $d(a)$ be its signature. For every pair a, b of vertices of c and every edge e from a to b, let $d(e)$ be the only edge between $d(a)$ and $d(b)$.

Limitations

As noted previously, the goal of the data discovery morphism is to reflect the relationships in the data domain, which are impossible to calculate, in relationships in the search domain, in which these relationships are, by definition ordinal and, therefore searchable. There are at least two factors that can make this reflection imprecise.

1. The signature extraction may not be injective. In this case, multiple datasets could be mapped to the same signature. Even though the signature generation algorithm generated exactly one signature for each dataset in the data domain, the signature could still represent more than one dataset. In this case, the DataBridge will consider these two datasets to be equivalent. Note that this may still be a useful result.

2. The similarity generated from m may be skewed from the true similarity in c. In this case, we would like a measure of how far away the generated similarity is from the true similarity. Still, as the true relationships are unknown, this computation is not possible. Nonetheless, it is desirable for signature algorithms to be unbiased, which is another way of saying we would like the mapping to be fair. The same fairness is assumed for the similarity algorithms. In our design, m is essentially an estimator of c, and we want a fair and unbiased estimator. Because m is a graph, the unbiasedness covers signatures and similarities.

The DataBridge in Action

The process of producing a searchable network of datasets using the DataBridge proceeds in several discrete phases. In this section, we introduce these steps. The remainder of this paper describes, in detail, each phase in the order of execution. Along

the way, we'll include some example algorithms for the framework, drawing these from the National Science Foundation-funded DBfN project (*DBfN: Data Bridge for Neuroscience: A novel way of discovery for Neuroscience Data*, 2012). Our primary partner in this grant is the SchizConnect (*SchizConnect: Effortless Data Collection*, 2019) project, an archive for storing data concerning schizophrenia. These steps include:

1. **Locate the datasets of interest.** Often these datasets will be found in one or more domain-specific repositories, but that is not a requirement.

2. **Extract or generate and store appropriate metadata.** This might involve scraping the web page for each dataset, interacting with SQL databases, or code that investigates dataset files. (morphism f)

3. **Create a Signature from the metadata.** Processing the raw metadata into a signature, enables the similarity algorithms to focus on what is most important in the metadata. (morphism g)

4. **Run a Similarity algorithm to produce a searchable, weighed, undirected network.** The choice of algorithm depends on the type of metadata and the nature of the data. (morphism h)

5. **Run a Social Network Analysis (SNA) algorithm to cluster the network results into dataset neighborhoods.** Social Network Analysis is an active research topic in Computer Science.

6. **Provide search and visualization capabilities.** Enabling researchers to find datasets is the endpoint of the entire process.

Locating Datasets

The first phase in building a searchable community of datasets is locating the data of interest. Without datasets, no data discovery system is of any interest. As we developed the ideas leading to the DataBridge proposal, we believed that we would work extensively on methods to obtain datasets and associated metadata from what Bryan

Heidorn (Heidorn, 2008) refers to as "Dark Data." Dark data is composed of datasets that, for several possible reasons, have become isolated and unavailable to anyone other than their originators (and sometimes not even to them). But in the last few years, the appearance of prominent domain-specific archives such as ClinicalTrials.gov, SchizConnect.org, and NOAA's National Centers for Environmental Information have allowed the work on the DataBridge to move forward without a significant effort towards retrieving Dark Data. Developing efficient methods for locating and integrating Dark Data continues to be an important challenge for scientific research and Data Discovery systems, but the rise in centralized data repositories removed that challenge from the DataBridge critical path. Instead, we are currently working in domains and with communities with well-defined dataset storage. We note that some repositories, such as Australia's Commonwealth Scientific and Industrial Research Organization (CSIRO) (*CSIRO*, 2019) now offer similarity-based discovery services for data in their archives.

Obtaining Metadata: Morphism f

Once the datasets of interest have been located, the next phase is obtaining actionable metadata for the datasets. At the heart of any data discovery system will be metadata about discoverable datasets. The DataBridge does not store any datasets; instead it only stores metadata about the datasets. Identifying, retrieving, and storing appropriate metadata is a fundamental challenge for data discovery systems. In the process of conceiving, proposing, and building the DataBridge, we have identified the following sources of metadata:

- Metadata associated directly with the data, by the data providers. For example, all of the datasets at ClinicalTrials.gov include a human readable and machine scrape-able text description of the entry.
- Metadata derived from the dataset. If the dataset is, for example, a netCDF file, the data field names in the file may be meaningful, especially if the field names have been chosen in accordance with a domain-specific ontology or controlled vocabulary such as the CF conventions (Gregory, 2003).

- User metadata, derived from monitoring human interactions with the datasets. If four different users have all examined and downloaded datasets A, B, C, and D, there is likely to be some exploitable relationship among these datasets.

- Method data, derived from the usage of datasets. If datasets A, B, and C, are all either the results of, or inputs to, the same computational model (e.g. the ADCIRC model used to solve ocean circulation and transport problems (*ADCIRC*, 2019)), these datasets are likely related in some way.

- Metadata derived from domain-specific analytics performed on datasets. Many times the results of executing an analytic on a dataset will be either a numeric or vector-based result that can be used as an input to a similarity routine. This technique, known as deep indexing, is potentially extremely powerful, as it allows indexing datasets whose content is entirely numeric and lacking useful metadata. An example of this technique is executing voxel-based morphometry analysis on brain images and using the results as a signature that can be input to domain-specific similarity algorithms. This will enable a new form of indexing for datasets that document the progress of health conditions such as Alzheimer's or Parkinson's disease.

The DataBridge platform utilizes a hierarchical schema for metadata storage, currently implemented using MongoDB, based on the DDI lite specification (*DDI Lite*, 2019). The metadata is segmented into individual namespaces as needed. A DataBridge namespace is a collection of datasets or metadata that are jointly the subject of analyses. All operations within the DataBridge are performed on a specified namespace. This allows the DataBridge to maintain separate networks for each domain community. Each namespace can have multiple associated networks, either because the similarity that produced the network has been executed more than once or because multiple similarity algorithms have been executed on the same namespace. As we'll see below, namespaces can also be created in the signature generation phase.

DBfN Example Each SchizConnect dataset has metadata associated with them in the form of the names of the various sets of questions (called an instrument) contained in each study. This metadata is contained in an Excel spreadsheet. We created code in Ruby that extracted the relevant metadata for each study into a single Json formatted file. We also created code that loads this metadata into the DataBridge persistent metadata storage.

Signature Generation: Morphism g

The next phase involves processing the raw metadata into a usable product. Just as every dataset has a collection of metadata that defines it to the data discovery system, every dataset has a signature used in the similarity analysis step. The signature and the metadata are sometimes identical, but a data discovery system often wants to pre-process the metadata before the analyses step. Some examples of this kind of processing from text metadata include:

- Removing "stop" words from text metadata. Stop words are common words in the language that are not germane to the domain and therefore are seen as not adding information to the signature. Words such as "at," "the," and "is" are typical examples.
- "Stemming" the metadata. Stemming involves reducing words to their most basic elements so that similar words are not mistaken for different words. Examples of this include removing pluralization (i.e., "cats" goes to "cat") and removing suffixes such as "ing" and "ly."
- Reduce the metadata to a set of "important" terms for comparison using, for example, term frequency-inverse document frequency (tf-idf) processing. Tf-idf processing is intended to find the most important terms in a corpus of documents. The goal of using tf-idf processing on some or all of the text metadata in a namespace is to produce a signature consisting of only the most significant terms.

The current DataBridge architecture does not store generated signatures as a separate

first-class object. However, if the code that loads the metadata also processes the metadata into a signature, that will be stored. Alternatively, since there is no restriction against representing the same dataset in multiple namespaces, generated signatures can be stored as additional metadata in a dedicated namespace. It's also possible to perform needed signature generation dynamically during the similarity analysis phase.

DBfN Example: We defined a signature consisting of a vector of boolean variables, indicating the presence or absence of an instrument in each dataset. Each dataset represented a study of neurological symptoms for a cohort of patients. Since each study used different terms for instruments that could represent similar concepts, the project team developed an ontology with a higher-level set of concepts to which the study terms are mapped. Each element in the signature vector represents one of these higher-level concepts. The signature routine uses the ontology to map the instruments in each study into the signature vector. In this case, we stored the original metadata and the signature in two different namespaces. While the original metadata is stored in the namespace SchizConnectRawMetadata, the signature used by the similarity algorithm is stored in the SchizConnectSignature namespace.

Similarity Analyses: Morphism h

In this phase, the metadata signatures are processed by a similarity algorithm to produce a similarity matrix, which is simply a weighted adjacency matrix. This is normally produced by executing the algorithm once for each pair of datasets in the namespace. In this matrix, each row and column represent one of the datasets in the namespace, and the value is a floating-point number that is the calculated similarity between the two datasets. A value of 0 indicates no relationship between the datasets, and 1 indicates that the datasets are indistinguishable by the current algorithm. The DataBridge stores this matrix in a network database (currently Neo4j), using an identifier for the similarity processing execution that refers to the metadata stored in the MongoDB.

Because the network is undirected and the similarity of any dataset to itself is 1, the matrix is only calculated for the upper triangle. This still requires order N-squared comparisons, so choosing efficient algorithms can be important as the number of datasets in a namespace increase. For example, consider that ClinicalTrials.gov now contains more than 250000 datasets, so a full pair-wise similarity analysis of this site would require order tens of billions of calculations. The DataBridge can exploit the fact that similarity operations are independent using a batch mode that provisions nodes on a computational cluster to carry out the calculations in parallel.

One less computationally intensive approach would be to develop a "reference" metadata for a namespace and then compare each dataset to the reference. This would have the substantial advantage of being order N. The DataBridge has not experimented with this approach, but the performance advantage will justify experimentation in the future, particularly as namespaces increase in size.

There is a wide variety of similarity analysis algorithms extent in the current literature. For text processing alone, the DataBridge project has implementations of more than a dozen algorithms. These are based primarily on the SimMetrics library from Simmetrics (*Simmetrics*, 2019). The DataBridge project has also implemented several variations of the Hamming Distance (Hamming, 1950a) algorithm. Still we have barely scratched the surface of the universe of possible algorithms, even for text metadata. The potential set of algorithms for processing non-text metadata is equally large. The important point is that the DataBridge can easily include whatever algorithm is appropriate for the metadata in the namespace.

DBfN Example Since the signature for the metadata in this project is a vector and all of the vectors are the same length, the similarity algorithm we use is a version of the Hamming Distance (Hamming, 1950b), modified to produce a similarity rather than a

distance. This algorithm compares every element in two signature vectors, counts the equivalent boolean entries, and divides the sum by the length of the vector. This calculation is done for each pair of signatures to produce the complete set of similarity calculations needed to complete the search domain. Figure 2 is a representation of this algorithm.

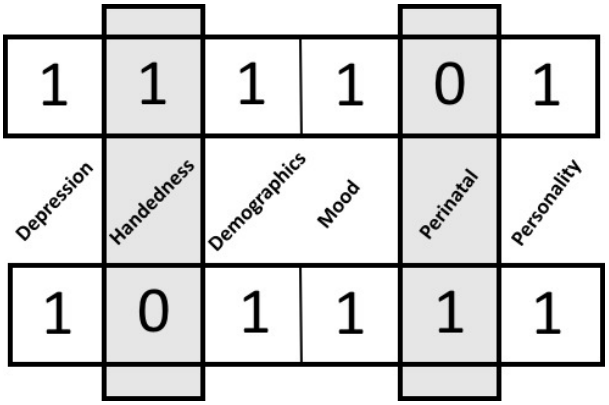


Figure 2. The Hammings Similarity

Social Network Analysis

The Social Network Analysis (SNA) phase is used to partition the datasets in the namespace into several possibly overlapping communities. The DataBridge stores the resultant partitioning as attributes of the adjacency matrix in the Neo4j network database. SNA is currently an active topic of research in Computer Science, and any reasonable overview of the current state of the art is well beyond the scope of this paper. SNA is important in data discovery systems because it provides insight into the structure and distribution of datasets in a namespace in a way that may lead to scientific insights that would otherwise be unmotivated. Additionally, partitioning the namespace allows more efficient searching and visualization, particularly in namespaces with many datasets. It may be possible to look at a graph with 100 nodes and extract some information, but this becomes much more difficult if the graph has 10,000 nodes. But a graph with 10,000 nodes portioned into a multilevel hierarchy with SNA analysis is much more tractable.

One important issue in considering possible SNA algorithms for data discovery, is that the similarity data produced in a pairwise comparison is not always a distance metric, in the sense that the similarities calculated will not satisfy the triangle inequality. The relationships of dataset A to dataset B and dataset A to dataset C say nothing about the relationship between datasets B and C. This fact rules out many of the common clustering algorithms, such as K-Means and DBSCAN. DataBridge has primarily used the clustering methods from the scikit-learn(Pedregosa et al., 2011) python-based clustering library. We have not yet implemented hierarchical clustering for visualization or search.

DBfN Example: As noted above, DataBridge relies on third-party providers for clustering algorithms. While there are several choices, the most commonly used algorithm, including for the DBfN project, has been the Agglomerative Clustering algorithm from sckit-learn. Note, however, that the Hammings distance measure we use for similarity in this project satisfies the triangle inequality, so the choice of potential SNA algorithms is broader than it might otherwise be.

Search and Visualization

Since the purpose of building a data discovery system is to help real users discover datasets of possible interest, efficient and robust searching is paramount. From the user's point of view, this can be as simple as a recommendation engine integrated into a domain-specific archive. In this case, once a user looks at a dataset in a network with closely similar datasets, the engine creates some form of a "Would you be interested in one of these?" message for the user. The DataBridge supports this type of interaction through "DataBridge as a Service," through which any existing user interface can implement a remote procedure call-style communication to query an existing DataBridge network for datasets similar to a dataset of interest to the user. There is a prototype DataBridge web application which allows users to execute this type of search.

As we've shown in section 3, our category theory justifies combining multiple *ddm* morphisms to enable more complicated queries. Because the DataBridge can maintain multiple networks corresponding to combinations of namespace and similarity algorithm, in principal the user should be able to ask for the most similar datasets using a combination of the various networks. This feature has not been implemented. Similarly, the user may want to interact with the network in a completely graphical manner, either, as mentioned above, to glean some knowledge about the entirety of the data space or because they find it a more efficient way to investigate the data networks. The DataBridge does have a simple visualization application that displays a clustered view of the network, but there is no hierarchy or search function, and it's not possible to display more than one network at a time, either side by side or overlaid.

Other Examples

In the introduction to this paper, we expressed our belief that all data discovery systems fit within the framework of Data Discovery Morphism. Having explained and justified that morphism, and illustrated it with the DBfN example, we briefly explain how two other systems fit into our framework.

The Netflix Example

As our first example, we consider the Recommender System (Gomez-Uribe & Hunt, 2015) Netflix uses to help users select content from an overwhelmingly large corpus of possibilities. Let us consider each video a dataset; while Netflix has many different parts of the user interface, let's consider the "Because You Watched" section. In this portion of the interface, the user is presented with suggestions stemming from one or more videos the user has previously streamed. This is analogous to the DBfN search capabilities described above, in which the user can find the datasets most closely related to one they already know. Netflix does not need to search for datasets, as the video content they provide is the core of their product, but to create content for the "Because You Watched" section, they do need metadata (Morphism f of our model). The

metadata they need comes from several sources we noted in our description of Morphism f. In addition to harvesting the metadata immediately available for each video, they also observe the user's interactions with the video, harvesting information such as the time of day, the day of the week and the intensity with which the viewer streams the video. A signature is generated (Morphism g) for each of these videos by combining the metadata sources. Details of the algorithms used to convert this metadata to a similarity ranking (Morphism h) are not, as far as we know, public, but Netflix does state that they use a variety of algorithms in their recommendation process, including both supervised and unsupervised learning approaches as well as matrix factorization.

The Neuroscience Information Framework

Our second example comes from the field of Neuroscience. The Neuroscience Information Framework (NIF) (Gardner et al., 2008) was designed to provide an integrated access point to as many neuroscience resources, including datasets, as possible. Unlike Netflix, the NIF project had to create an inventory of datasets. In this process, they realized they needed a commitment from dataset owners not only to provide access to the data but to create and share metadata by annotating the datasets appropriately. This process is, of course, Morphism f of our model. In addition to the metadata requirements, the NIF team realized that they would need to develop a common set of terms to describe, among other artifacts, the datasets and their contents. This was done with a series of expert terminology workshops. Once applied to the individual datasets, these terms enable the creation of signatures for the datasets, thus conforming to our Morphism g. At first glance, it may not seem that NIF has implemented Morphism h. Instead, they have provided a concept-based query interface that can search multiple resources. But this type of search still conforms to Morphism h. To see this, note that the resources that the search returns are by definition similar. They fulfill the query parameters and, at least for the query domain, have an implied similarity of 1. All of the other unselected resources have an implied similarity of 0. In effect, each individual query is a distinct similarity algorithm, an application of

Morphism h, which produces a matrix comprised entirely of ones and zeros.

Remaining Challenges and Conclusion

Data Discovery systems are in their infancy and will doubtless assume increasing importance in the next few years as the data deluge facing domain scientists and data consumers deepens. Here are a few of the most pressing challenges:

- **Evaluation:** Developing metrics for data discovery system effectiveness will be a critical challenge. Tangible outcomes such as the number of recommended data set downloads, the number of queries answered, and the number of data discovery system cites in scientific papers are fairly easy to develop. But evaluating the relative effectiveness of the various similarity measures, let alone the possible combinations of similarity measures when there is no ground truth against which to measure, will continue to be difficult.

- **Scale:** The largest network for which the DataBridge has executed pairwise similarity algorithms is approximately 18000 datasets. Compute time was order 100 hours, which is not a significant burden for a modern academic cluster. However, attempting to scale the network by a factor of 100 to 1.8 million datasets would, with no improvements in efficiency, require order 1 million CPU hours. Similarly, SNA algorithms for large networks may become infeasible without significant parallelization. Developing and evaluating effective, ideally order N, similarity and parallel SNA algorithms will allow continued scaling. In particular, developing either similarity algorithms that satisfy the triangle inequality or techniques for converting the existing similarity data to satisfy the equality would allow a wider selection of SNA algorithms.

- **Search:** Developing and evaluating the most useful ways for researchers to interact with networks and SNA partitions of much larger networks of datasets will be difficult. This could be particularly important if research shows that combining multiple network views and multiple partitions on the same community of datasets is useful.

- **Inclusiveness:** One possibility for data discovery systems is to increase their domain beyond a single domain or group of related domains to try to encompass larger and larger swaths of scientific data. Such work might enable us to discover new connections between previously isolated scientific disciplines. As science becomes increasingly multi-disciplinary, data discovery systems will be motivated to follow suit.

As data volumes scale, many in the data science community have increasingly recognized the need for effective discovery systems. The DataBridge was one of the first efforts to meet this challenge. Our experience building the DataBridge led us to develop a formal theory of how data discovery proceeds and create a category theory-based justification for our discovery process. While we have made significant progress toward our initial goals, we believe a great deal of research and development is yet to come in this field.

ACKNOWLEDGMENTS

The efforts described in this paper are funded by the NSF Office of Cyberinfrastructure OCI-1247652, OCI-1247602, OCI-1247663 grant, “BIGDATA: Mid-Scale: ESCE: DCM: Collaborative Research: DataBridge - A Sociometric System for Long Tail Science Data Collections”, (2012-2015) and by the NSF IIS Div Of Information & Intelligent Systems grant number #1649397 “EAGER: DBfN: Databridge for Neuroscience: A novel way of discovery for Neuroscience Data” (2016 – 2017). We are also grateful to Lei Wang and Kathryn Alpert of the Feinberg School of Medicine, Northwestern University and Jessica Turner of Georgia State University for their work on the DBfN project. We thank our friend and co-investigator Tom Carsey for his many contributions to the DataBridge project.

References

ADCIRC. (2019). Retrieved from `http://adcirc.org`

Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval* (Vol. 463). ACM press New York.

Berry, M. W., & Martin, D. I. (2005). Principal component analysis for information retrieval. In *Handbook of parallel computing and statistics* (pp. 415–430). Chapman and Hall/CRC.

Big Data: DataBridge. (2019). Retrieved from `http://databridge.web.unc.edu/`

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Borceux, F., & Stubbe, I. (2000). Short introduction to enriched categories. In B. Coecke, D. Moore, & A. Wilce (Eds.), *Current research in operational quantum logic: Algebras, categories, languages* (pp. 167–194). Dordrecht: Springer Netherlands. Retrieved from `https://doi.org/10.1007/978-94-017-1201-9_7` doi: 10.1007/978-94-017-1201-9_7

Coecke, B. (2006). Introducing categories to the practicing physicist. in: What is category theory. In *Advanced studies in mathematics and logic 30, pp.45–74, polimetrica publishing.*

CSIRO. (2019). Retrieved from `https://www.csiro.au/en/About`

DBfN: Data Bridge for Neuroscience: A novel way of discovery for Neuroscience Data. (2012). Retrieved from `https://www.nsf.gov/awardsearch/showAward?AWD_ID=1649397`

DDI Lite. (2019). Retrieved from `http://www.ddialliance.org/specification/ddi2.1/lite/index.html`

Deerwester, S., Dumais, S., Landauer, T., Furnas, G., & Beck, L. (1988). Improving information-retrieval with latent semantic indexing. In *Proceedings of the asis annual meeting* (Vol. 25, pp. 36–40).

Dumais, S. T. (2004). Latent semantic analysis. *Annual review of information science and technology*, *38*(1), 188–230.

Fuhr, N. (1992). Probabilistic models in information retrieval. *The computer journal*, *35*(3), 243–255.

Gardner, D., Akil, H., Ascoli, G. A., Bowden, D. M., Bug, W., Donohue, D. E., ... Williams, R. W. (2008). *The neuroscience information framework: A data and knowledge environment for neuroscience.* doi: 10.1007/s12021-008-9024-z

Gomez-Uribe, C. A., & Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems*. doi: 10.1145/2843948

Gregory, J. (2003). The cf metadata standard. *CLIVAR Exchanges*, *28*.

Hamming, R. W. (1950a). Error detecting and error correcting codes. *Bell System Technical Journal*, *29*, 147–160.

Hamming, R. W. (1950b). Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, *29*(2), 147–160. doi: 10.1002/j.1538-7305.1950.tb00463.x

Harris, D. R. (2015). Modeling Reusable and Interoperable Faceted Browsing Systems with Category Theory. In *Proceedings - 2015 ieee 16th international conference on information reuse and integration, iri 2015* (pp. 388–395). doi: 10.1109/IRI.2015.65

Heidorn, P. B. (2008). Shedding light on the dark data in the long tail of science. *Library Trends*, *57*(2), 280–299. doi: http://doi.org/10.1353/lib.0.0036

Luo, C., Zhan, J., Xue, X., Wang, L., Ren, R., & Yang, Q. (2018). Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International conference on artificial neural networks* (pp. 382–391).

MacLane, S. (1971). *Categories for the working mathematician.* New York: Springer-Verlag. (Graduate Texts in Mathematics, Vol. 5)

Nguyen, Long. (2015). *Topic modeling with more confidence: a theory and some algorithms.* URL: `http://dept.stat.lsa.umich.edu/~xuanlong/Talks/B_admix_pakdd.pdf`.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of*

Machine Learning Research, *12*, 2825–2830.

Rajasekar, A., Kum, H., Crosas, M., Crabtree, J., Sankaran, S., Lander, H., ... Zhan, J. (2013). The DataBridge. *Science Journal*, *2*, 1. doi: ISBN:978-1-62561-001-0

Rajasekar, A., Sankaran, S., Lander, H., Carsey, T., Crabtree, J., Kum, H., ... Zhan., J. (2013). Sociometric methods for relevancy analysis of long tail science data. In *Ase/ieee international conference on big data.*

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation.*

Robertson, S., & Zaragoza, H. (2009). *The probabilistic relevance framework: Bm25 and beyond*. Now Publishers Inc.

Robertson, S. E., & Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society for Information science*, *27*(3), 129–146.

Rosen, R. (1958). The representation of biological systems from the standpoint of the theory of categories. *The Bulletin of Mathematical Biophysics*, *20*(4), 317–341. doi: 10.1007/BF02477890

Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

SchizConnect: Effortless Data Collection. (2019). Retrieved from `http://schizconnect.org`

Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, *18*(3), 491–504.

Simmetrics. (2019). Retrieved from `https://github.com/Simmetrics/simmetrics`

Singhal, A., et al. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, *24*(4), 35–43.

Spivak, D. I. (2009). Simplicial Databases. *Arxiv preprint arXiv:0904.2012*, 1–35. Retrieved from `http://arxiv.org/abs/0904.2012`

Tang, J., Meng, Z., Nguyen, X., Mei, Q., & Zhang, M. (2014). Understanding the limiting factors of topic modeling via posterior contraction analysis. In

International conference on machine learning (pp. 190–198).

Vickers, P., Faith, J., & Rossiter, N. (2013). Understanding visualization: A formal approach using category theory and semiotics. *IEEE Transactions on Visualization and Computer Graphics*, *19*(6), 1048–1061. doi: 10.1109/TVCG.2012.294

Yu, C. T., & Salton, G. (1976). Precision weighting—an effective automatic indexing method. *Journal of the ACM (JACM)*, *23*(1), 76–88.