Looking at Everything in Context

Zachary G. Ives[1] Zhepeng Yan[1] Nan Zheng[1] Brian Litt[2,3] Joost B. Wagenaar[3]
Departments of Computer & Information Science[1], Bioengineering[2], and Neurology[3]
University of Pennsylvania
Philadelphia, PA, USA
{zives,zhepeng,nanzheng}@cis.upenn.edu, littb@mail.med.upenn.edu, joostw@seas.upenn.edu

ABSTRACT

The database field has increasingly broadened past carefully controlled, closed-world data, to consider the much more complex space of data resources on the Web. In this area of "open" Web and contributed data, there are vast quantities of raw data — but there is a limited understanding about real or realistic *usage scenarios and problems.*

In turn, this has made it difficult to assess the effectiveness of data integration and structured search techniques. To take the next step, data integration researchers need to pool their resources around a small number of cloud-hosted "hub" applications with real users, to gain access to the sorts of workload-driven evaluations that are only possible in industry today. We present our early work on the HABITAT system, an extensible data hosting and management platform for *evaluating integration techniques in situ.* We describe an initial deployment in a neuroscience setting, including "mid-term" lessons learned in building the platform and community. We conclude with a set of research challenges that we believe will be instrumental in accelerating open data integration.

1. INTRODUCTION

The data management world has changed substantially over the past decade: users have higher expectations for being able to query and analyze data, yet the data increasingly comes from *open* (dirty, inconsistent, incomplete, heterogeneous, often unstructured) sources, such as direct user contributions or Web crawls. Efforts to solve *open data integration* ("Web data integration," "dataspaces" [20], the "Semantic Web" [8], "knowledge graphs" [10], etc.) aspire to help users *query across relationships* instead of just within documents.

To address these new challenges, there has been a blurring of the lines dividing databases, search, natural language processing, and knowledge representation, as well as increasing emphasis on crowdsourcing and semantic data extraction and integration. Examples of algorithmic and systems innovations in recent years include modeling of uncertainty [42], automated Web-scale data extraction [13, 16, 44], new techniques for schema and entity alignment [17, 19, 41], and reasoning about provenance, responsibility, and trust [28, 36]. New visions have been proposed to develop *pay as you go* integration [20, 43], and to incorporate human (crowd) components into querying [21].

Yet, this progress on algorithms and methodologies has seldom translated into end-to-end solutions to integration problems. Perhaps the biggest issue is that we only have a limited understanding of the problems we need to solve. Instead of the traditional TPC-H-style closed-world with answers based on logical expressions, with measures that can be precisely computed from the data — we have a world of approximate-matching metrics that must be tailored to data characteristics, and ranked query answers whose utility often must be evaluated *with respect to the user's goals.* Yet today we typically can only conjecture about integration scenarios and user goals.

The result of all of this is that many fundamental questions remain unanswered. We do not know what the "killer applications" are for open data integration, nor what levels of query expressiveness and semantic integration are necessary for the common use cases. There are many potential ways of designing the layers of an integration solution (extraction [49], matching [41], query answering [45]) that produce approximate results; and of using ranking strategies or machine learning techniques to compose these [7, 47]. Techniques are being proposed to explore using human and automated approaches together [21, 47]. Yet we have little guidance about what works well in practice.

We have great *data* resources on the Web: Freebase [10], DBpedia [2], YAGO [44], GenBank [38]. However, other than perhaps research challenges like the Open Ontology Alignment Initiative (`oaei.ontologymatching.org`) and the TREC Knowledge Base Acceleration effort (`trec-kba.org`), we lack a good picture of how they would *actually be used.*

1.1 Missing Capabilities from Academia

Google, Facebook, and Microsoft (Bing) *do* understand the usage patterns of their platforms, by virtue of their scale and their direct interface to users. Instead of generating synthetic workloads over available data, these companies monitor real users, searches, and data. This view *at scale* yields many benefits [25]:

- **Network effect**. Many resources are only useful if there is enough adoption. Consider the success of Wikipedia and Facebook, and the challenges that alternative platforms have had in replicating their success. Many academic efforts, even if they leverage real data,

7th Biennial Conference on Innovative Data Systems Research (CIDR'15) January 4–7, 2015, Asilomar, CA USA.

have the problem of limited visibility and usage.

- **Usage patterns**. Both in terms of data and users, industry has insight into what data is being processed, and how people are using it. Simple frequency analyses of term co-occurrences, click-throughs, etc. are very effective in identifying what matters. Machine learning becomes useful once there is training data.
- **A/B testing and field experiments**. As described in [4, 14, 23, 48], Web companies do extensive A/B testing to measure the impact of certain features. Such testing methodologies, at sufficient scale, provide evidence-based means of assessing ranking functions, alignments, data presentation methods, etc.
- **Surveys and polls**. Another challenge in academia tends to be finding representative subjects for user studies or surveys. Thus some researchers turn to Amazon Mechanical Turk to conduct certain kinds of studies [35]. However, an existing user base can be a more representative and informative population.

Such capabilities help inform what problems are most important (the "80% solutions"), show where the bottlenecks are, and identify what provides the most benefit. Google, Bing, and Facebook credit many improvements to usage-driven evaluation. It seems clear that the open data integration research area needs similar capabilities (and also that industry is not yet directly focused on this subfield).

1.2 Academic "Living Labs"

A key question is *how can we acquire the types of insights available at a Google or a Facebook from academia*, while focused on problems at the "bleeding edge". Can our community create "*living laboratories*," which enable researchers to collaborate with real users (doing real work) in evaluating their novel algorithms, metrics, interfaces, and tools? We argue that the state of technology and science have advanced to the point is indeed *yes*, if we are strategic. We describe our steps towards this long-term vision:

- We identify promising application domains in need of community data sharing solutions, which might provide a lens into actual behavior (Section 2.1).
- We describe our early work on the HABITAT platform, which has been designed not to solve a specific set of data integration needs, but rather to enable *in situ* evaluation of alternative techniques developed by the research community (Section 2.2).
- We describe our initial lessons learned in specializing the HABITAT platform towards neuroscience research, through the `IEEG.org` portal (Section 6).
- We propose a set of challenge problems that must be addressed to further flesh out the vision of a series of community-specific "living laboratories" for data management research (Section 7).

Our hope is that our HABITAT platform, as well as a small number of others strategically positioned in communities that can make use of open data integration techniques, might serve as the foundation of a new generation of data integration research backed by user and workload-driven studies.

2. LIVING LABS FOR DATA INTEGRATION

Over the past decade, our group has been involved in a variety of data integration and sharing efforts with academic (generally scientific) collaborators [29, 40]. Each effort has faced many obstacles in attempting to use it as an evaluation domain for computer science research techniques.

A major challenge is ensuring that the *user community can be productive in their own areas of focus*, while offering computer science researchers the opportunity to *incorporate new experimental features* that could be evaluated. This requires an initial long-term investment in developing core functionality, and constant effort in promoting the effort.

Unfortunately, a requirement from the computer science perspective is that the effort achieves significant scale. If one is to achieve the vision of a "living laboratory," this requires adoption by a large enough user community, who has data as well as expertise in analyzing the data. Moreover, it must be possible to get a bird's eye view of the users, data, and code (using a hosted solution, like Google or Amazon RDS, as opposed to providing a tool like PostgreSQL).

We believe that this goal is now achievable for the first time. Many research areas with interesting data are *suddenly re-orienting into data-centric fields*: for instance, consider the recent changes in neuroscience (with the BRAIN initiative), or the emerging areas of network science and the movement in sociology towards Web- and digital social network-based user studies. We discuss these and other promising areas in Section 2.1.

Moreover, with the advent of modern cloud-hosted services and tools, and successful open source or user-contributed models, it becomes feasible to build, host, and operate community-scale software and services. Consider how Eclipse and Linux provide core support for a wide variety of different sub-projects, and how GitHub and StackOverflow serve tremendous numbers of collaborating users. A prerequisite, however, is a core platform that can be customized to the needs of the application domain and to the needs of database researchers. We describe our prototype platform, which we call HABITAT, in Section 2.2.

2.1 Finding a Foothold

Finding the right "foothold" applications is a significant challenge. Our own experience has been that the most critical aspect is strong collaborator in the target domain, who is a leader and influential figure in his or her field. Building novel and useful tools is not enough; one needs visibility and connections to foster platform adoption and sustainability. An additional consideration is that it is generally easier to gain traction in a discipline with little data management infrastructure, versus one with a large legacy base. In either case, however, a great deal of work must be invested in changing people's habits (see Section 6.2).

To drive data integration research, there is also a requirement for some minimum level of diversity, complexity, and open-endedness to the problems. Standard file formats, traditional keyword search, and hand-coded queries are adequate solutions for certain classes of applications. Database-style solutions lend themselves towards answering queries in a "heavy tailed distribution": large numbers of queries that are individually seldom-asked. Thus we must seek the latter properties.

Even with these filter criteria, there are many possible "foothold" applications through which data management re-

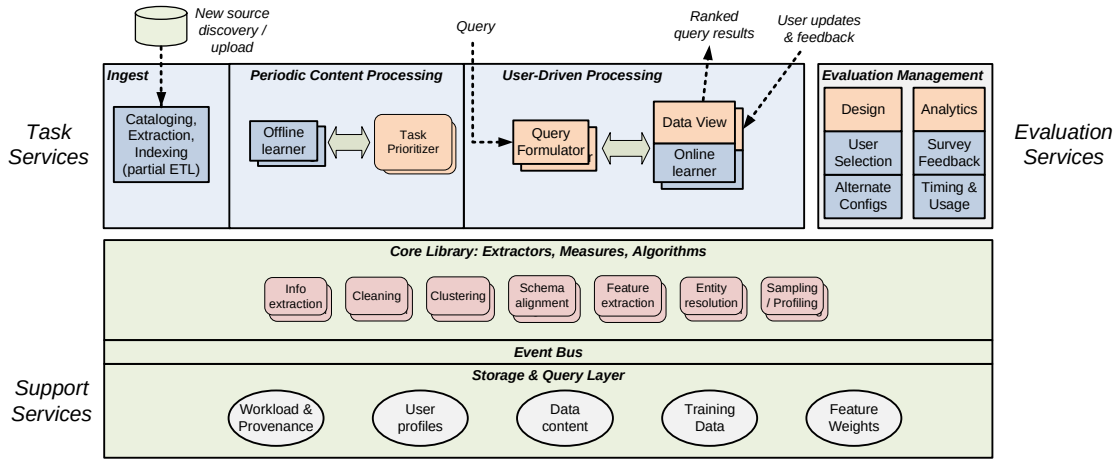

Figure 1: Habitat architecture diagram

search might be driven, particularly within biomedical and social science fields that are just beginning to focus on data-driven methodologies. Several promising domains include neuroscience (Section 6), the intensive care unit and hospital emergency room, cardiology, wearable or implantable devices, and emerging efforts to bring "big data" techniques to the social and behavioral sciences. The main ingredients are a reasonable amount of digital data, a community consensus that data analysis can yield new insights, and a strong collaborator who can co-lead the effort from within the domain. Additionally, the ideal situation is that researchers (both from the database field and the target domain) will focus on a small number of central collaborative efforts, rather than a multitude of competing platforms.

2.2 The Habitat Platform

We have previously constructed numerous data integration systems [26, 28, 31], as well as several biomedical applications. Based on this experience, we have been developing a unified platform that can be easily customized to serve the needs of various application domains — yet also supports new research at any layer of the data integration stack, possibly even extending to computer-human research (e.g., crowdsourced computation, collaborative filtering), self-tuning research (e.g., more efficient storage systems), and methods for visualization and querying of data.

Our early prototype HABITAT platform (Figure 1) is built to accept a set of "pluggable" components at different levels; this is in contrast to typical database systems that provide integrated solutions. Our goal is a repurposable framework for interconnecting various third-party-provided modules, much like Eclipse does today for software development tools. One of our key focal points is the interaction between the data integration *platform* (the area in which the database community is most comfortable) and the user-facing *software*.

HABITAT modules should be possible to mix and match for different target domains with different constraints — and it should be feasible to configure them to do *in situ evaluation* of alternative strategies, e.g., A/B testing [4]. To achieve scale and elasticity, the platform is designed to be cloud-hosted, using standard cloud building blocks (Web application servers, key-value stores, hosted RDBMSs, load balancers, etc.).

For flexibility we adopt a "pay as you go" data integration model [20, 43] that heavily builds upon our recent Q System [46, 47, 51]. Our platform can store (or link via the Web to) data of any type, e.g., as objects within a key/value store. However, the amount of query functionality will depend on how much extraction, translation, processing, and linking has been applied to the data. These processing steps may occur over time, and may be directed by humans or algorithms.

Our main objective is flexibility, with performance a secondary goal that we address on an as-needed basis, after conducting real evaluation. We adopt a very general graph data model, where nodes represent data items and edges represent (possible) links. In contrast to RDF and other similar data, each node and edge in our platform (as with our Q System [47]) is associated with a set of *features* that can be used to derive a score. These features can include data provenance, scores from various matchers or recognizers, attribute values from the data, and so on. The algorithm for computing the scores, and for returning top-k ranked results, can be customized for the application.

HABITAT is designed to operate as a cloud-based Platform-as-a-Service (PaaS) with additional, configurable, Software-as-a-Service (Saas) front-end modules for search, visualization, and evaluation. We divide the architecture of HABITAT into three service categories. Support Services are shared across system components and invoked via a common interface. Task Services initiate processing of the data. Finally, the Evaluation Management Service allows us to configure the other components, possibly on a per-user or per-experiment basis.

3. SUPPORT SERVICES

The Support Services provided by the HABITAT platform are illustrated in the bottom tier of Figure 1. They encompass the storage layer, communications mechanisms for modules, and the library of operators that can be applied to manipulate or parse the data.

3.1 Storage Layer

The middleware **Storage Layer** takes a variety of raw data types (binary files, text, time series, tuples, XML, JSON, hyperlinks, graph data) and stores them according to a configurable policy. In our existing platform, there are multiple underlying storage systems, including a large-object key-value store (currently Amazon S3 or the filesystem), a small-object

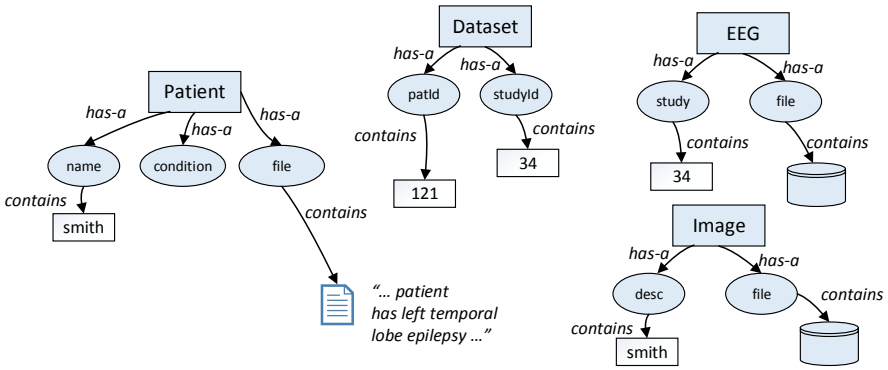

Figure 2: Example content graph upon load of several database tables and text documents. This virtual graph is partitioned across several subsystems in the Storage Layer. Each edge and node is also annotated with a set of *features* (not shown) from which quality or relevance scores will be derived.

key-value store (MongoDB), and a relational DBMS. We allow for *links* across the various data storage subsystems, such that the data is abstractly a single *content graph* in which both nodes and edges can have weighted *features*. The model and storage formats extend those we developed for the Q Query System [32, 46, 47, 51].

EXAMPLE 1. *See Figure 2 for a simplified content graph for neuroscience data uploaded by users. Perhaps one data provider uploads patient metadata in the form of a* PATIENT *table that references a clinician's PDF document. Another uploads a collection of EEG traces in a file format called MEF, which includes header information that about the associated dataset; as well as a set of records for the* DATASET *table. A third source provides* IMAGE *data. Through pay-as-you-go data integration techniques developed in the Q System [46, 47] and implemented in the Task Services layer (Section 4),* HABITAT *should ultimately discover links among the provided data.*

Beyond encoding the core data of interest, the storage layer also records *provenance* for derived data. Provenance comes in a variety of forms. For database-style views and queries, we capture derivations using a variant of the semiring provenance formalism, encoded in relations [24, 33]. We augment this style of provenance with in formation about which results were observed by the user, and any user feedback or actions on the data items. Of course, much of scientific data processing goes beyond database-style queries to "black box" data functions (e.g., in MATLAB, Python, R, or Java). Here, we use a traditional graph representation of inputs and data flows [6, 34, 39]. Ultimately we hope to use a combination of the ideas from ProQL [33] and Lipstick [1] to allow for complex reasoning and link analysis over the provenance (and associated log records). We believe that there are many opportunities for harnessing the data provenance and usage information to make quality assessments and recommendations [30].

Another key task of the Storage Layer is to maintain collections of data used for algorithm *training*. These are generally curated by data experts, who additionally create gold standard annotations.

Finally, the Storage Layer maintains assigned weights for various features that contribute to the scoring model. As we describe in the next section, these weights can be customized per user and collaborative filtering techniques [50] can be used to customize them across the user community.

3.2 Event Bus

In contrast to a standard database system, HABITAT is very focused on third-party *extensibility* of its core functionality. We envision that our team or other collaborators will continue to come up with new modules that enable visualization of different datatypes, modules that enable the creation of custom survey forms or usage modes, and so on.

To facilitate this, we borrow an abstraction commonly used in extensible event-driven software: the *Event Bus*. Modules, particularly those that interact with the user, can register themselves as *listeners* (using the standard event-condition-action paradigm) when an event occurs. Various system services trigger *events* whose descriptions are sent as messages along the event bus. Typical events include the registration of a new dataset to the "Ingest" module of Section 4, a successful user login, or the completion of an analysis job. The Event Bus is also used to coordinate requests across distributed deployments of HABITAT.

3.3 Core Library of Operators

Finally, HABITAT contains a library of typed *operators*, which are strongly typed in terms of their inputs and outputs. The **Core Library** includes:

- *Extractors* and file format readers, which allow us to "pull" semantic content from files or complex data structures, or to determine (machine learning-style) *features* that affect the score of a result. In general, we assume that multiple different extractors may be applicable to any given file or data object.
- *Measures* compute a semantic distance between items, such as schema elements or data values. These commonly represent matchers from schema matching [22, 41] and record linking [19]. In HABITAT we use a variety of off-the-shelf algorithms from COMA++ [15] as well as our own custom algorithms [46]. In principle, any clustering coefficient could also be used, although we have not yet investigated this possibility.
- Finally, we allow for *algorithms* (clustering, sampling, annotation, scientific data analysis) that condense, connect, or produce sketches of data. These will generally

be domain-specific, and triggered by the end-user or a custom plugin module.

Each operator typically creates new nodes and/or edges that are added to the content graph. Such outputs are annotated with features indicating the operator(s) that produced them, enabling the system to associate a reliability or confidence weight with each operator. The Core Library's operators are used within the task-oriented subsystems of the platform.

4. TASK SERVICES

We divide the main task services into an *ingest* service for handling new content, a *periodic content processing* service that provides prioritized background processing of existing data, and an *online processing* service that handles user queries, updates, and external API usage. We describe each of these components in more detail.

4.1 Ingest

Experience has shown that users will be more willing to contribute data if they are not required to do any conversion or processing on their end. Hence any user upload triggers an event on the Event Bus. In turn this ultimately invokes the *ingest* service, a variant of the standard ETL (extract/transform/load) pipeline used in many data warehousing settings. The Ingest stage uses typing information to identify any relevant workflows that perform tasks related to data extraction, transformation, quality analysis, and so on.

Of course, it is not necessarily the case that we have a full set of ETL tools to convert any user's data into a unified form. Thus we emphasize an extensible, incremental ingest process. As each data collection is uploaded, we store it as a collection of binary objects, along with available metadata about the collection.

If the appropriate file format readers are available, the Ingest stage will read the data out and create new nodes and edges in the content graph of the Storage Layer, annotated with features that will result in confidence scores. It additionally indexes any textual content using an inverted index.

4.2 Periodic Content Processing

Rather than assuming a set of regularized stages that take raw data and convert it into warehoused data, we adopt a more general model that allows for *iterative refinement* of the data content and system operation over time.

The periodic content processing module is given a set of *event-action-condition* rules that poll the data to see if further processing can be done. Each *action* consists of a sequence of Core Library operators, e.g., to extract content from a BLOB of a known file type, to run a data quality measure, or to better link the data in the system by running entity resolution or schema matching algorithms. Note that as the system is running, further operators may be registered with the Core Library, thus increasing the set of potential transformations or operations that can be invoked.

Of course, the space of potential operators may be much larger than the set of computational resources we are willing to devote. Given a budget, the **Task Prioritizer** attempts to determine which operations to apply next, and to which subsets of the data. It can make use of a machine learning component (Offline learner) and usage history to determine what algorithms and data values are of most importance. Our Task Prioritizer is currently under development, but we see it as a key aspect of iteratively improving the integration level of the data integration platform. See Section 7 for more details on some of the challenges that must be solved.

4.3 Online Processing

We assume that any user, administrator, or external interaction with the system yields two effects: the user or external request is handled, and evaluation information is gathered to improve the operation of the system. The **Online Processing** module implements these actions.

Here, the user asks for a particular data resource using some particular interface or modality (e.g., Web form, text search, Web service API). The Query Formulator will turn this into a form that can be answered from the Storage Layer. Our current emphasis is on structured keyword queries, for which we leverage the techniques developed in the Q Query System [46, 47, 51]. Here, *search-driven integration* techniques combine on-demand schema matching with structured keyword search [9, 27].

EXAMPLE 2. *Figure 3 illustrates how in the search-driven data integration model of the Q System [46], a keyword query posed by the user over the original content graph of Figure 2 will be handled. Initially, the user's query (for EEG and image data for patients whose condition is left temporal epilepsy) is encoded as a set of keyword nodes in the graph (dark nodes at the bottom). From there, similarity measures are used to map the terms to nodes representing values, attributes, and relations (see unlabeled arcs between keyword nodes and other nodes). Then a series of iterative* expansion *steps are performed: for each keyword node, schema matching and record linking algorithms are invoked between the* frontier nodes *connected to the keyword nodes, and other nodes in the content graph; if these exceed a threshold new edges are added, expanding the frontier (see* LINKSTO *edges added to the original graph). Eventually a set of top-k Steiner trees are returned*[1].

Query results will appear in a **Data View** appropriate for the datatype. Structured search resutls are presented in a tree-style hierarchical browser, whereas custom viewers are invoked for images, time series data (see Figure 4), and so on. Within the set of search results, the user (or user's external tool) may browse, manipulate, and edit the data.

User interactions and timings are immediately broadcast on the Event Bus and also logged to the Storage Layer. An *online learner* may use that feedback to directly update the query and results as the system interacts [46, 51]. Our online learner again comes from the Q System: here, an algorithm called MIRA takes user feedback on the relative scores of output results, and adjusts the weights of individual features to match the feedback. This learning step enables the system to learn how much to weigh the output of each source, extractor, matching tool, etc., and thus to improve the quality of its query results.

5. EVALUATION MANAGEMENT SERVICE

[1]More precisely, we use the output of a top-k Steiner tree approximation algorithm [47].

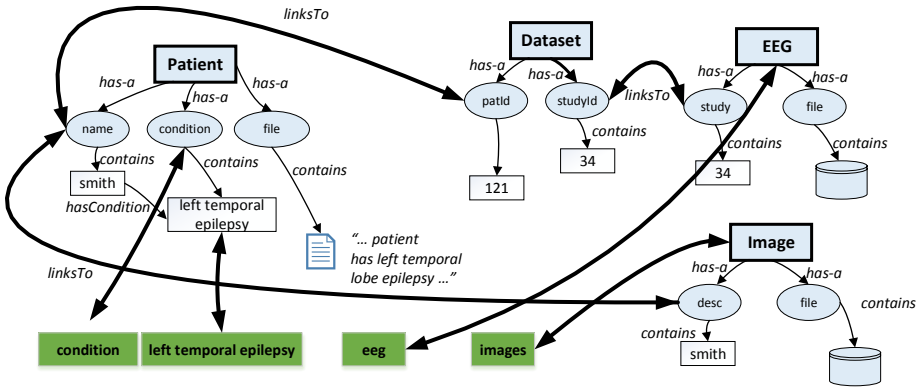

Figure 3: Example content graph as a query is posed and different edges are computed.

Beyond providing core services for supporting user actions, we also want to allow data integration researchers to evaluate alternative algorithms, metrics, and modules against one another *in situ*. The final service focuses on experiment design, followed by evaluation.

Our goal is to enable researchers to identify target user groups, by querying their profile, history (e.g., datasets contributed, publications), or various workload parameters, and can send these users consent requests or surveys. Moreover (as in [4]), they may specify different implementations of components (e.g., viewers, matchers, etc.) that are deployed to different user or user groups, such that A/B testing can be done to compare results using normal activities.

Unlike with Facebook [4] or Google, in our setting the person conducting the experiment is not necessarily a member of the development group responsible for the core platform: rather, the experiment might be conducted by a user who has developed a "plug-in" module and should not have direct access to the raw data, query logs, and so on. We are developing a set of options that enable the study participant to grant different levels of data and query visibility to the user conducting the study. For instance, rather than presenting the search terms and data as part of the evaluation results, we might instead present aggregate quality measures, or substitute hashed values that maintain some properties (e.g., value-equality testing) without revealing the underlying data.

Of course, underlying these capabilities, we are also building query services for the logs, user response timings, and user feedback histories. We believe this core set of capabilities will provide a depth of validation that has, to this point, not been achievable in open data integration.

6. PILOT DEPLOYMENT: IEEG.ORG

The previous section describes our core, domain-independent HABITAT platform. However, the platform has been co-developed alongside a targeted application, which seeks to provide data integration and big data capabilities to the neuroscience research community. To this point, neuroscience has suffered from a lack of standardization and shared infrastructure for addressing many of its grand challenges, e.g., understanding neurodegenerative diseases.

Starting with the domain of epilepsy research and treatment — where there is a need for algorithmic techniques to *detect* the onset of seizures in order to build implantable

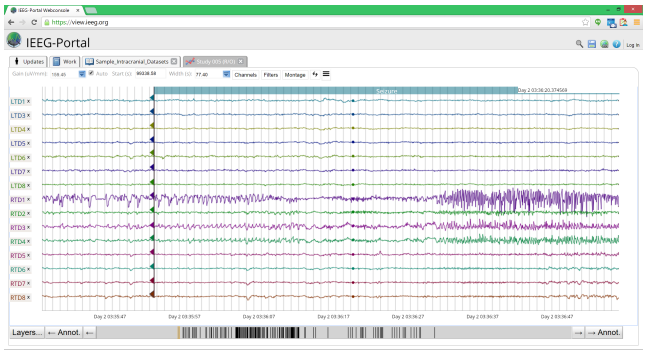

Figure 4: IEEG.org deployment of Habitat for neuroscience

medical devices that can administer electric shocks to *preempt* the seizures — we have been specializing HABITAT in the form of a platform called IEEG.org, which consists of an HTML5-based Web data portal, a set of APIs, and a many-TB data repository. IEEG.org's domain goals are to enable neuroscientists to search for and share time series and imaging data for patients and animals with epilepsy; to build and evaluate new data analysis algorithms for epilepsy research; and to collaborate in the production and analysis of this data. Our second set of goals revolve around fleshing out and validating the HABITAT platform.

6.1 Customization for the IEEG Platform

IEEG.org specializes HABITAT along several dimensions.

Storage layer. IEEG features a combination of relational, text, time series, imaging, and hierarchical data. Our storage layer encompasses a series of open-source (MySQL, MongoDB, BerkeleyDB), cloud (Amazon S3), and custom services to hold this data. Time series datasets often feature 10s-100s of electrode channels, each sampled in the 10s of KHz for days or months. Building upon a compressed file format called the Multiscale Electrophysiology Format [12], we developed new time series subsystems for storing and streaming the data, "pushing down" signal processing operations like downsampling and filtering to the earliest possible points.

Access Control and Security. IEEG.org was focused on

community-wide data sharing, which is often viewed as a "structured Wikipedia for the field." Yet a critical feature was support for *access control* because many users wanted to support *small* collaborations that would run for a time, before results were published and then data was published. Access control often encompasses not only the source data and operations being applied, but even search logs: scientists are often concerned that others will determine what they are doing and "scoop" them. Some industry partners are even concerned about controlling access to derived results, necessitating a model in which permissions associated with a derived dataset are always a *subset* of those associated with the parent dataset.

Periodic Content Processing. A major challenge for IEEG is that each new source of content is often from a different neuroscience sub-community, with not only different data formats but different means of recording metadata. There is typically a mix of raw time series and imaging data as well as clinical metadata, annotations, case histories, etc. We separate the content ingestion step (where data is uploaded by users directly to Amazon S3, and added to a pending queue) from a deferred content processing stage.

During the content processing stage, we perform data integrity checking and conversion as appropriate to the file format; we run a data quality reports; we pre-index all structured and textual data; and if feasible, we map the metadata into our core schema. The various processing steps are specified as pipelines built up from modules in the **Core Library**, triggered via event-condition-action rules. As new modules or pipelines are added, these will automatically be applied to old as well as new data.

Online Processing. A lesson learned in building the user-facing aspects of the IEEG.org effort was to focus on maximizing end-user productivity, while keeping the computer science research at the periphery.

While the data integration aspects of most interest lay in building *universal search* across uploaded, indexed, partly integrated data using techniques from our Q System [47, 51], we also constructed traditional Web query forms for the more common query patterns. A major point of emphasis was providing an interactive data viewer for time series (EEG) data that operated in the browser in real-time (see right side of Figure 1). Over time, other viewers for different data modalities such as MRI images — and capabilities for switching views — have become essential.

Core Library, Web Services, and Virtual Machines. Most neuroscience researchers are more comfortable writing their own custom data analysis code than reusing existing modules. Hence a second point of emphasis was on a set of APIs to make it feasible to connect such algorithms to our platform and to register these within the Core Library in a seamless way. In order to support scale-out, we developed MapReduce-like APIs to handle time series data *en masse*. However, our early users were more comfortable writing single-threaded, main-memory, procedural code in MATLAB. Thus the MATLAB usage pattern became a significant point of emphasis in the early stages. Ultimately, as the user community has grown over time to encompass industry partners, we are again returning to use cases that require a broader treatment of data-parallel analysis using clusters of virtual machines. In some cases, due to access control restrictions, we have had to provide custom virtual machines that have no network access to the external Internet.

Collaborative Tools. Social networking and data management are typically viewed as disjoint capabilities. However, since IEEG.org needed to serve as a platform that engaged users and facilitated science, we developed a variety of collaborative, social-network-like features around data, users, and tools. Discussion boards, messaging facilities, project teams, usage notifications, and data provenance tracking became the means of connecting users, tools, and data — and of maintaining engagement.

It is too early to declare victory. However, IEEG.org currently has 560 users and 1250 contributed datasets, is a key part of several community-wide NIH and DARPA efforts, and serves as a gateway between large biomedical device companies and the neuroscience research community. The effort has taught us several lessons that we believe are of broader significance to open data integration research efforts.

6.2 Midterm Lessons Learned

Computer scientists are not representative

We have found a significant difference in culture between computer science and the life sciences. Computer scientists are typically problem-driven, and expected to frequently come up with new problems and generalizable solutions. Small variations in our assumptions often produce different results. Hence, except in industry settings, there is relatively low risk to open-sourcing or publicly posting our data and code resources. Thus our culture has embraced these practices.

In contrast, much of experimental science is driven by (expensive-to-collect) data and proprietary methods. When sharing data and methods, a scientist is giving up a significant first-mover advantage, which in turn means it is much harder to convince scientists to share data.

■ *Successful collaborative tools and techniques from computer science do not necessarily extend to other sciences.*

"Passive sharing" is a major hurdle

Abstractly, every scientist agrees that data sharing is important, and moreover, funding agencies or journals often mandate publication of the data. However, many individuals engage in "passive sharing" by posting their data to satisfy their obligations, but they make no effort to ensure others can actually make use of the data. Key assumptions may be missing, file formats may be under-specified, etc. (To be fair, at times supporting others is a time-consuming task with few rewards.)

■ *We must offer incentives that are viewed as outweighing the risks involved, which also means we must be able to measure the factors we wish to incentivize.*

Public data does not create data users

More challenges arise when attempting to cultivate a community of *data users*. In areas like astronomy or genomics, where a small number of shared instruments were collectively used by a broad array of researchers — not only does information management infrastructure exist for the scientists, but there is a large community of researchers who never

directly interact with the instruments, but instead look for data on the Web. Unfortunately, in less infrastructure-rich communities like neuroscience, there exists no separate community of data scientists, so the availability of data does not necessarily translate into a set of data users.

■ *Lack of a data scientist culture makes it hard to recruit users, as we must convince many individuals to use shared as well as local data.*

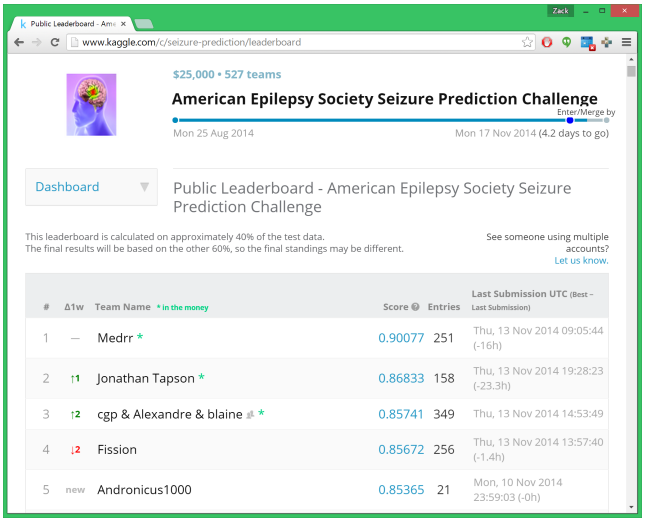

Figure 5: Leaderboard for Seizure Prediction Challenge on kaggle.com, ongoing as of 17-Nov-14.

Visibility requires sustained effort

Part of the challenge to getting engaged users is, in fact, to get users in the first place. Standard tactics — going to conferences, hosting workshops — are effective to some extent, but in fact, for the IEEG.org effort, our greatest recruiting tool has proven to be a contest aimed at data scientists who were not necessarily in our original target neuroscience community.

Here, we were able to bootstrap off `Kaggle.com`, a company that helps corporations and organizations run data science contests. For the Penn-Mayo Seizure Detection Challenge (`kaggle.com/c/seizure-detection`), we curated several datasets hosted on IEEG.org, and formulated a contest (with a cash prize) to build the best epileptic seizure-detection algorithm. The contest was heavily publicized and drew 205 participants, many of whom submitted entries on a daily basis. The state of the art in seizure detection has been relatively primitive for some time; but as of the time of this paper (just before contest closing), the top-scoring entry has achieved a detection accuracy of over 97% (with the top-12 entries scoring above 95%). The visibility and success of the contest has already drawn many users to our endeavor, and some of these users are making use of IEEG.org. We are in the midst of conducting a second challenge (sponsored by the American Epilepsy Society) for seizure *prediction* in streaming fashion, at `kaggle.com/c/seizure-prediction`. Figure 5 shows the leaderboard as of mid-November 2014. Visibility attained from the first challenge, as well as official sponsorship and a larger prize, have increased participation to 527 teams.

■ *Challenge problems with prizes are an important means of engagement and help create success stories. An open question is how to maintain visibility and engagement for such challenges, once many groups are competing.*

Sustainability requires a long-term plan

Software development is expensive and time consuming, and involves much more than a few graduate students connecting together their research projects. Funding agencies now worry about sustainability, as most platform efforts wither away beyond the initial effort. Our strategy has been to "sprint" to develop the core platform along the lines of Figure 1; then later to adapt this platform to serve both database research (e.g., our search [51] and large scale data analysis [37] efforts) as well as a variety of new user communities. Those separate efforts require much less development time and produce much faster payoff, hence a slew of them can jointly sustain the overall effort. Our hope is that efforts like HABITAT can also be used to bootstrap future projects with similar goals.

■ *The core platform requires dedicated funding, but thereafter it may be feasible to sustain via multiple specialization efforts.*

The world keeps changing

Our experience with the project affirms the notion that *incremental* approaches to standardization, which are more agile in handling new use cases, are essential. Through the first few years of the IEEG.org effort, not only did our target community change several times (clinician-epileptologists who looked at human data, neuroscientists who also looked at animals, implantable device manufacturers), but the kinds of data and usage scenarios also shifted and expanded. Our basic platform and our software engineering practices had to keep changing towards models that were easily extensible, including by third parties.

■ *The pay-as-you-go integration approach, where data is iteratively improved over time, is essential in handling scale. So is a platform which supports rapid adaptation.*

7. RESEARCH CHALLENGE PROBLEMS

As our effort continues to move forward, we have identified a number of critical research challenges that we believe are bottlenecks towards progress in open data integration.

Structured data sharing metrics and incentives

A major challenge moving forward, across the data-centric sciences, is determining the metrics — then later, the rewards — for sharing data. A key question is *what is the equivalent of the h-index (the "s-index") for data*?

Given data in a usage (provenance) graph, a starting point would be measures like ObjectRank [5] or eigenvector centrality [11] to assign scores to the data, possibly based on where the "uses" of the data appear in the literature [30]. However, this approach assumes that data is atomic, whereas in reality much scientific data is both joined and aggregated to produce results. A challenge moving forward is understanding how to prorate the impact of the data here, possibly using a proportional model like that of *responsibility* [36]. Conversely, there are a set of questions about how to combine the impact scores of many individual data items to rate the repository or the user.

Strategic improvement of the data

To achieve the vision of pay-as-you-go data integration, we believe it is essential not only to react to user changes — but also to *proactively* work on improving the data (hence our *periodic content processing* step). As the volumes of data on the platform grow, and the Core Library supports many different extraction, matching, and clustering steps, the system (whether purely algorithmic or augmented by the crowd) does not have the resources to do all processing. It must focus on the highest-payoff processing tasks: what data to align algorithmically, what cleaning tasks to perform, when to solicit human attention to improve a portion of the data, etc. The actual payoff may not come until multiple steps are combined. Hence, we need to look at how to build a system that *plans a set of actions* that *have the highest anticipated cost-benefit payoff*. Traditionally the database field has only looked at optimization and planning in a very specialized way; in the future we are likely to need a more sophisticated AI planning mechanism.

Privacy-preserving user experiments

User studies are challenging even when data is not personally identifying and subjects can be asked for consent (i.e., HIPAA considerations are met). Many scientific users want their queries, data, and usage patterns to be kept confidential, at least until they have published results. To this point, the experiments we have conducted on IEEG.org have only included code from us or third-party code that is *trusted*, and we have not looked at the actual data (only the performance differences). Our users know that their workloads and activities can be used in computer science experiments, but that such data will remain confidential. This largely mirrors the terms of usage at Facebook [3, 4, 23], Microsoft [14], and Google [48], where extensive A/B testing and workload monitoring are done.

To reach the broader vision of platforms where third-party researchers can contribute their own (semi-trusted) code to the platform and conduct experiments — the challenge is how to ensure that data and metadata are adequately protected. Can we guarantee, e.g., that running a contributed schema matching algorithm will not reveal the data being matched? The full guarantees of, e.g., differential privacy [18], may not be necessary for public data — but it seems essential to develop techniques for validating that the algorithms are implemented in a data-independent fashion (hence do not recognize specific data), and ensuring they are applied to data with enough diversity and/or noise to ensure their output is meaningful yet privacy preserving. Alternatively, they must be restricted to data for which consent was obtained.

It is worth noting that our goal with IEEG.org is to also facilitate sharing of data that might *itself* be private. Here, with respect to allowing third-party experiments, there are both technical questions with stricter constraints than described above (what privacy assurances can we give, perhaps now requiring differential privacy-style guarantees), as well as broader legal and administrative policies (will Institutional Research Boards give consent to such experiments). It remains unclear whether this more ambitious setting will be feasible, but clearly it presents interesting problems.

8. CONCLUSION

The field of open data integration will remain immature until enough "living labs" are available to help guide and evaluate research problems and solutions. In this paper, we have proposed a unifying architecture for platforms that facilitate both applied science and computer science — via a combination of data integration, collaborative tools, and management of user-based evaluations. We have presented our experiences in implementing and evaluating these concepts in the IEEG.org platform, and proposed a series of open research challenges that are essential towards solving the greater challenges of the field.

Acknowledgments

This work was funded in part by NSF IIS-1217798, NIH U24-5U24NS063930, and gifts from Google and Amazon.

References

[1] Y. Amsterdamer, S. Davidson, D. Deutch, T. Milo, J. Stoyanovich, and V. Tannen. Putting lipstick on a pig: Enabling database-style workflow provenance. In *Proc. VLDB*, 2011.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.

[3] E. Bakshy and D. Eckles. Uncertainty in online experiments with dependent data: an evaluation of bootstrap methods. In *KDD*, 2013.

[4] E. Bakshy, D. Eckles, and M. S. Bernstein. Designing and deploying online field experiments. In *WWW*, 2014.

[5] A. Balmin, V. Hristidis, and Y. Papakonstantinou. ObjectRank: Authority-based keyword search in databases. In *VLDB*, 2004.

[6] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: Enabling interactive multiple-view visualizations. *IEEE Visualization*, 2005.

[7] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis. Keyword search over relational databases: a metadata approach. In *SIGMOD*, 2011.

[8] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[9] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, 2002.

[10] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[11] P. Bonacich. Simultaneous group and individual centralities. *Social Networks*, 13, 1991.

[12] B. H. Brinkmann, M. R. Bower, K. A. Stengel, G. A. Worrell, and M. Stead. Multiscale electrophysiology format: An open open-source electrophysiology format using data compression, encryption, and cyclic redundancy check. *Proc. IEEE Eng Med Biol Soc.*, 2009.

[13] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the web. *PVLDB*, 1(1), 2008.

[14] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham. Seven pitfalls to avoid when running controlled experiments on the web. In *KDD*, New York, NY, USA, 2009. Available from `http://doi.acm.org/10.1145/1557019.1557139`.

[15] H. H. Do and E. Rahm. Matching large schemas: Approaches and evaluation. *Inf. Syst.*, 32(6), 2007.

[16] A. Doan, R. Ranakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, and W. Shen. Community information management. *IEEE Data Engineering Bulletin*, December 2006.

[17] X. L. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In *VLDB*, 2007.

[18] C. Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12, Venice, Italy, July 2006. Available from `http://research.microsoft.com/apps/pubs/default.aspx?id=64346`.

[19] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE TKDE*, 19(1), 2007.

[20] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34(4), 2005.

[21] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: answering queries with crowdsourcing. In *SIGMOD Conference*, 2011.

[22] A. Gal. *Uncertain Schema Matching*. Synthesis Lectures on Data Management. Morgan and Claypool, 2011.

[23] A. Grant and K. Zhang. Airlock: Facebook's mobile A/B testing framework. Technical report, Facebook, 2014. `https://code.facebook.com/posts/520580318041111/airlock-facebook-s-mobile-a-b-testing-framework/`.

[24] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Update exchange with mappings and provenance. In *VLDB*, 2007. Amended version available as Univ. of Pennsylvania report MS-CIS-07-26.

[25] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2), 2009.

[26] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *ICDE*, March 2003.

[27] H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: ranked keyword searches on graphs. In *SIGMOD*, 2007.

[28] Z. Ives, N. Khandelwal, A. Kapur, and M. Cakir. ORCHESTRA: Rapid, collaborative sharing of dynamic data. In *CIDR*, January 2005.

[29] Z. G. Ives, T. J. Green, G. Karvounarakis, N. E. Taylor, V. Tannen, P. P. Talukdar, M. Jacob, and F. Pereira. The ORCHESTRA collaborative data sharing system. *SIGMOD Rec.*, 2008.

[30] Z. G. Ives, A. Haeberlen, T. Feng, and W. Gatterbauer. Querying provenance for ranking and recommending. In *TaPP*, 2012.

[31] Z. G. Ives, A. Y. Halevy, and D. S. Weld. Adapting to source properties in processing data integration queries. In *SIGMOD*, June 2004.

[32] M. Jacob and Z. G. Ives. Sharing work in keyword search over databases. In *SIGMOD*, 2011.

[33] G. Karvounarakis, Z. G. Ives, and V. Tannen. Querying data provenance. In *SIGMOD*, 2010.

[34] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 2006.

[35] W. Mason and S. Suri. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior Research Methods*, 44(1):1–23, 2012. Available from `http://dx.doi.org/10.3758/s13428-011-0124-6`.

[36] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *PVLDB*, 4(1), 2010.

[37] S. Mihaylov, Z. G. Ives, and S. Guha. REX: Recursive, delta-based data-centric computation. In *PVLDB*, 2012.

[38] National Center for Biotechnology Information. GenBank. Available from `www.ncbi.nlm.nih.gov/GenBank/`.

[39] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10), 2006.

[40] Processing phylOData (pPOD). http://phylodata.seas.upenn.edu/cgi-bin/wiki/pmwiki.php.

[41] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4), 2001.

[42] C. Re, N. N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[43] A. D. Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD*, New York, NY, USA, 2008.

[44] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A large ontology from Wikipedia and WordNet. *J. Web Sem.*, 6(3), 2008.

[45] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. 2011.

[46] P. P. Talukdar, Z. G. Ives, and F. Pereira. Automatically incorporating new sources in keyword search-based data integration. In *SIGMOD*, 2010.

[47] P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. G. Ives, F. Pereira, and S. Guha. Learning to create data-integrating queries. In *VLDB*, 2008.

[48] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer. Overlapping experiment infrastructure: More, better, faster experimentation. In *KDD*, New York, NY, USA, 2010. Available from `http://doi.acm.org/10.1145/1835804.1835810`.

[49] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*, 2012.

[50] Z. Yan and Z. G. Ives. Collaboratively learning to repair links. Submitted for publication, 2014.

[51] Z. Yan, N. Zheng, Z. Ives, P. Talukdar, and C. Yu. Actively soliciting feedback for query answers in keyword search-based data integration. *PVLDB*, 2013.