DBLife: A Community Information Management Platform for the Database Research Community

[Demonstration]

Pedro DeRose[1], Warren Shen[1], Fei Chen[1], Yoonkyong Lee[2], Doug Burdick[1], AnHai Doan[1], Raghu Ramakrishnan[3]

[1]University of Wisconsin [2]University of Illinois [3]Yahoo! Research

1. INTRODUCTION

Community Information Management: There are many communities on the Web. Some are based on common interests, such as communities of movie goers, database researchers, and bioinformaticians, while others are based on a shared purpose, such as organization intranets and online technical support groups. Community members often want to discover, monitor, and query entities and relationships in their community. For example, database researchers might want to know if there is a connection between two given researchers, where a given paper has been cited in the past week, or what of interest has happened in the last 24 hours.

Answering such questions often requires retrieving *raw, largely unstructured* data from multiple sources (e.g., home pages, DBLP, mailing lists), then inferring and monitoring semantic information. Examples of such inference and monitoring include recognizing entity mentions (e.g., "J. Gray", "SIGMOD-04"), deciding if two mentions (e.g., "J. Gray" and "Jim Gray") refer to the same real-world entity, recognizing that a relationship (e.g., co-authoring, advising, giving a talk) exists between two entities, detecting new entities (e.g., new workshops), and inferring that a relationship (e.g., affiliation with a university) has ceased to exist. The above inference and monitoring tasks are well known to be difficult [1, 2, 3, 7, 10]. As online communities proliferate, developing effective solutions to support their information needs becomes increasingly important. We call this problem *community information management*, or CIM for short.

The Cimple Project: To address the CIM problem, we have recently started Cimple, a joint project between the University of Wisconsin and Yahoo! Research [4]. Our goal is to develop a software platform that a data-rich online community can quickly deploy and customize to effectively manage its data. This software platform can be valuable for communities in a broad range of domains, ranging from scientific data management [5], government agencies, PIM [2, 8], dataspace management [6], and enterprise intranets, to those on the World-Wide Web. Cimple's approach has three steps: (1) We start with a high-quality seed provided by a community expert; this seed includes relevant data sources and domain knowledge about entities and relationships of interest. (2) We exploit this seed using simple but focused automatic methods to create and maintain an *entity-relationship* graph of the community. (3) We leverage the community by providing valuable, carefully crafted functionalities whose use helps correct, maintain, and evolve this graph.

In general, Cimple attempts to extend the footprints of DBMSs and more broadly apply database technologies to manage Web data. A major problem with doing database and IR research that involves the Web is that the Web is simply too big. In academic environments, it is possible, but difficult, to build infrastructures and user bases at the Web scale to uncover more interesting problems and better validate solutions. Cimple can be viewed as attempting to circumvent this problem by focusing on Web communities, which are in effect "mini-Webs". At this scale, it may be the case that it is easier to build infrastructures and user bases, to perform deeper semantic analysis to infer more complex structured data, and to apply database/IR technologies.

The DBLife System: To drive and validate research in Cimple, we are building DBLife, a prototype system that manages information for the database research community (see dblife.cs.wisc.edu). Eventually we may want to build more prototypes for research communities, such as AILife and IRLife, as well as non-research ones, such as those for the legal community and the community of movie goers.

DBLife has been live on the Web for about 1.5 years. DBLife currently monitors nearly 900 data sources, and downloads 9,500 pages, or 150+ MB, daily. It tracks roughly 335,000 mentions of 16,600 entities, and provides a variety of services that exploit the generated entity-relationship graph, including a daily community newsletter, entity *superhomepages* (pages that aggregate all detected and inferred information about an entity), and community event tracking.

We have briefly introduced DBLife at a SIGMOD-06 tutorial, and will demonstrate DBLife at CIDR-07. For the demonstration we will (a) motivate the CIM problem and Cimple project, (b) showcase a variety of DBLife's features, (c) explain the working of DBLife's internals, and (d) illustrate a range of open research issues in CIM. We currently plan to release a beta version of DBLife in January 2007. If so, the demonstration will showcase both the features of the released system, as well as the novel features under development. We now provide more details about DBLife and the

demonstration.

2. DEMONSTRATION OVERVIEW

2.1 CIM and Cimple

The first part of the demonstration will briefly describe and motivate the CIM problem and the Cimple project, as above.

2.2 DBLife Features

Next, we will demonstrate DBLife's features. Specifically, we will demonstrate the community daily newsletter, superhomepages of researchers, mass collaboration features, and community event tracking pages. We will also demonstrate features that are scheduled to be finished by the time of the demonstration (see below).

Newsletter: DBLife's newsletter, show in Figure 1, displays interesting events the system has inferred on a given day. These events are inferred using simple information extraction rules that exploit structural elements in carefully chosen data pages. For instance, DBLife can infer when a researcher is mentioned as a PC member in conference homepages, when a call for papers is announced on DBWorld, when a researcher is giving an invited talk, and many other such events. We will also demonstrate the provenance features of DBLife that are related to the newsletter.

Superhomepages: DBLife creates a page for each entity (e.g., organizations, researchers, publications) that aggregates all detected or inferred information about that entity. For example, Figure 2 shows the superhomepage of Divesh Srivastava. The main block of the page lists all discovered mentions of him in reverse chronological order. Along the right-hand side, it displays general information, followed by inferred relationships, services, and other events, such as talks and tutorials.

Mass Collaboration: Researcher superhomepages include a row of images related to that researcher along the top. This is an example of DBLife's mass collaboration. The images are gathered by searching on an image search engine, and users vote on which images should not be related to the researcher. Their votes are processed by a mass collaboration system [9], and over time the incorrect images are weeded out.

Event Trackers: As mentioned above, DBLife infers events using information extraction rules that exploit document structure. These events not only appear in the newsletter and superhomepages, but also in separate tracking pages. Such pages exist for conference information, paper acceptances, and invited talks. A sample of the invited talks page is shown in Figure 3.

New Features: DBLife is under constant development. Examples of features we are currently developing include context-sensitive mass collaboration features, personalization, entity-aware search, stronger provenance, and an integrated feedback framework.

2.3 DBLife Internals

In this part of the demonstration we briefly describe DBLife's general architecture, then the core workflow that generates and maintains the ER graph.

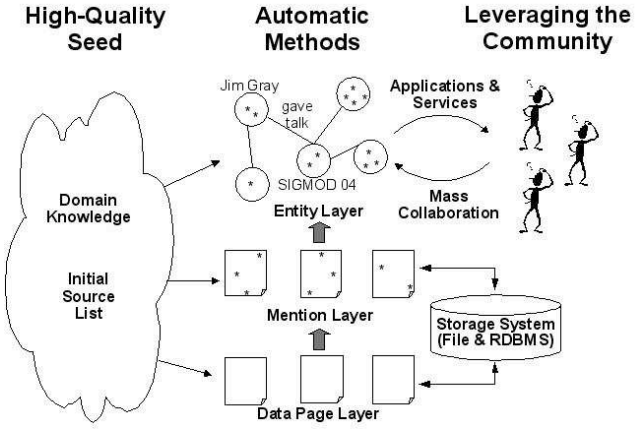

Figure 4: DBLife workflow

2.3.1 Architecture

DBLife comprises an easily extensible set of independent but intercommunicating modules. Each module takes as input XML files and outputs XML files accessible by other modules. A central configuration file specifies the modules' input and order of execution; when the system is run, the configuration file is interpreted, and the modules are executed in turn. After each execution, their output is archived so that it can be accessed at any point in the future.

DBLife's modules are divided into two groups. The first, called the *core modules*, is responsible for creating and maintaining the system's ER graph; it is described in more detail below. The second group, called the *application modules*, exploits the ER graph to generate output displayed by DBLife's web interface. These application modules are the heart of DBLife's features, and their functionality is described above.

2.3.2 Workflow

DBLife follows the Cimple approach of starting with a high-quality seed, building on this seed with automatic methods, and filling gaps left by these methods through mass collaboration. We now demonstrate DBLife's workflow, illustrated in Figure 4, in the context of the Cimple approach.

High-Quality Seed: As prescribed by Cimple, DBLife starts with information supplied by a community expert. This seed includes an initial list of sources to crawl, such as researcher home pages, conference pages, and the DBWorld mailing list. It also includes domain knowledge, such as entities and relationships of interest, and hints for extracting and maintaining them. For example, the expert provides a dictionary of entity names, gathered from DBLP, which DBLife uses to find entity mentions as explained below. The domain knowledge in this initial seed is used throughout DBLife's automatic methods, as illustrated in Figure 4.

Automatic Methods: Next, DBLife uses automatic methods, implemented as core modules, to create and maintain the ER graph. These methods rely heavily on domain knowledge, which is currently worked into the algorithms, but will eventually be expressed separately through a declarative language. DBLife's core modules are loosely organized into three layers: the data page, mention, and the entity layers. The modules in these layers run daily as dictated by the central configuration file.

Modules in the data page layer crawl the specified sources, cache downloaded pages, and provides access to previously

Figure 1: DBLife's front page, featuring the daily newsletter

Figure 2: A researcher's superhomepage

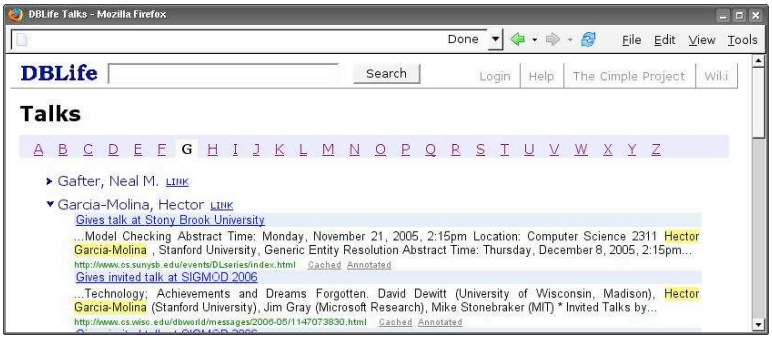

Figure 3: The event page for invited talks

cached pages. A set of modules extracts metadata for each page, such as when it first appeared and last changed. Other modules detect structural elements within pages, from small constructs, such as lists of proper names, to classifying entire pages, such as calls for papers on DBWorld. These structures are used to specify domain knowledge, such as rules for filtering mentions, inferring relationships, and detecting events as described in Section 2.2

The next layer handles entity mentions. First, modules use a dictionary of names supplied by the community expert to find mentions, represented as asterisks in Figure 4. Next, mentions from the current day are reconciled with those from the previous day: modules detect new mentions, and map old mentions to those from the previous day. This *mention tracking* is vital for maintaining the ER graph, since detecting community changes requires telling old mentions from new. This problem is difficult because it requires accurately determining the context of a mention within a page. Finally, a set of modules finds metadata for each mention, such as when it first appeared.

Last is the entity layer. It matches mentions to one another, disambiguates them, and groups them into entities. It then infers relationships between entities using domain knowledge rules, thus creating an entity-relationship graph. Finally, similarly to mentions, the current set of entities is reconciled with those from the previous execution: we must determine that the "Jim Gray" entity from today is the same as the "Jim Gray" from yesterday. This *entity tracking* problem is also difficult, since not only do the mentions comprising an entity change, but the real-world entity itself also changes. For example, when a researcher changes affiliations from university U to company C, we must still recognize him as the same person, even though his old mentions will be in the context of U while his new are in the context of C.

Leveraging the Community: Since automatic methods are inherently imperfect, DBLife leverages the community through mass collaboration to correct errors, and to help maintain and evolve the system's ER graph. However, using mass collaboration requires that the user have an incentive, and also enough information to help. Current DBLife features are a first step in this direction, and planned future features will further address these issues.

For incentive, DBLife provides functionalities that are valuable to the user, but also carefully designed so that their use helps identify or address issues within the system. An example is the image feature on superhomepages: users who wish to see the correct picture for a researcher have an incentive to weed out incorrect ones. Another example, which is still being developed, will allow users to directly edit certain parts of their superhomepages. Since users have a vested interest in keeping their superhomepages accurate, they are likely to help correct errors and contribute information.

To gain trust and insure quality feedback is provided, DBLife will provide users insight into the inference process through two primary mechanisms. First, *provenance* for all system inferences will be made available to the user. Examples include tracking the extraction rules used to generate a mention, and the logic used to group these mentions into entities. Second, DBLife will provide a representation of the *uncertainty* in the inference results. For example, the system may be aware that output from different mention extraction rules have different precision. This could be reflected using *confidence scores* assigned to the mentions to reflect estimates of their quality. Leveraging user feedback to update these estimates is an important area of future research. The uncertainty representation must also capture the *structure* of the uncertainty. For example, how does information that two mentions are co-referent affect other mention grouping decisions? Uncertainty management for DBLife is currently under active development.

2.4 Open Research Issues

We will leverage DBLife to illustrate the broad range of open research directions in CIM. Examples include how to perform large-scale information extraction efficiently; how to integrate information extraction and RDBMS technologies; how to explain the extracted data to the user; how to construct provenance, explanation, and uncertainty mechanisms that work for such contexts; how to perform better disambiguation of inferred data; and how to maintain the extracted data as the underlying raw data evolves.

3. REFERENCES

[1] P. Andritsos, R. J. Miller, and P. Tsaparas. Information-theoretic tools for mining database structure from large data sets. In *SIGMOD*, 2004.

[2] Y. Cai, X. L. Dong, A. Y. Halevy, J. M. Liu, and J. Madhavan. Personal information management with semex. In *SIGMOD*, 2005.

[3] W. Cohen. Information extraction. Tutorial, www.cs.cmu.edu/wcohen/ie-survey.ppt, 2003.

[4] A. Doan, R. Ramakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, and W. Shen. Community information management. In *IEEE Data Engineering Bulletin, Special Issue on Probabilistic Databases*, volume 29, 2006.

[5] J. Gray, D. T. Liu, M. A. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber. Scientific data management in the coming decade. *SIGMOD Record*, 34(4), 2005.

[6] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, 2006.

[7] T. Johnson and T. Dasu. Data quality and data cleaning: An overview. In *SIGMOD*, 2003.

[8] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A general-purpose information management tool for end users based on semistructured data. In *CIDR*, 2005.

[9] R. McCann, A. Kramnik, W. Shen, V. Varadarajan, O. Sobulo, and A. Doan. Integrating data from disparate sources: A mass collaboration approach. In *ICDE*, 2005.

[10] S. Sarawagi. Graphical models for structure extraction and information integration. Keynote talk and tutorial at ICDM-05, 2005.