KBB: A Knowledge-Bundle Builder for Research Studies

David W. Embley[1⋆], Stephen W. Liddle[2], Deryle W. Lonsdale[3], Aaron Stewart[1], and Cui Tao[4]

[1] Department of Computer Science,
[2] Information Systems Department,
[3] Department of Linguistics,
Brigham Young University, Provo, Utah 84602, U.S.A.
[4] Mayo Clinic, Rochester, Minnesota 55905, U.S.A.

Abstract. Researchers struggle to manage vast amounts of data coming from hundreds of sources in online repositories. To successfully conduct research studies, researchers need to find, retrieve, filter, extract, integrate, organize, and share information in a timely and high-precision manner. We are building a system that implements active conceptual modeling for learning as a way to give researchers the tools they need to perform their tasks in a more efficient, user-friendly, and computer-supported way. The system involves (1) creating "knowledge bundles" (KBs), which are data representations that enable researchers to perform their information extraction and organization work, and (2) providing a "knowledge bundle builder" (KBB) to help researchers develop KBs in a synergistic and incremental manner. The KBB can support both individual and team work, and can even help provide for larger-scale research repositories to be shared widely among the research community.

1 Introduction

In many domains, the volume of data is enormous and increasing rapidly. Unfortunately, the information a researcher requires is often scattered in various repositories and in the published literature. Researchers need a system that can efficiently locate, extract, and organize available information so they can analyze it and make informed decisions.

As a specific example, to do a recent study about associations between lung cancer and TP53 polymorphism, bio-researchers needed to: (1) do a keyword-based search on the SNP data repository for "TP53" within organism "homo sapiens"; (2) open each returned page one by one and find those coding SNPs that had a minor allele frequency greater than 1%; (3) for each qualifying SNP, record the SNP ID and many properties of the SNP; (4) perform a keyword search in PubMed and skim the hundreds of manuscripts found to determine which manuscripts were related to the SNPs of interest and fit their search criteria;

⋆ Supported in part by the National Science Foundation under grant #0414644

and (5) extract the information of interest (e.g., statistical information, patient information, and treatment information) and organize it. Doing high-precision information gathering for this cancer research study took a huge amount of time and effort. [5] How can a system help do high-precision information gathering for this kind of research study—indeed for any intelligence-gathering research in any scientific, business, military, or government domain?

We address this challenge with the idea of a *Knowledge Bundle* (*KB*) and a *Knowledge-Bundle Builder* (*KBB*). Active conceptual modeling for learning (ACM-L) is at the core of our approach. As we explain below, a KB includes an extraction ontology, which allows it to both identify and extract information with respect to a custom-designed schema. (This constitutes the conceptual-modeling part of ACM-L.) Construction of a KB itself can be a huge task—but one that is mitigated by the KBB. Construction of the KB under the direction of the KBB proceeds as a natural progression of the work a researcher does to manually identify and gather information of interest. As a researcher begins to work, the KBB immediately begins to synergistically assist the researcher and quickly "learns" and is able to take over most of the tedious work. (This constitutes the active-learning part of ACM-L.)

In describing our KBB approach to building KBs, we first give a bio-research scenario to show how our KBB can help bio-researchers harvest and manage data for a bio-research study (Section 2). We then explain how the KBB plays its claimed role in the bio-research scenario (Section 3). Finally, we give the status of our implementation and mention current and future work needed to enhance KBs and the KBB (Section 4) and then draw conclusions (Section 5).

2 Bio-Research Scenario

Suppose a bio-researcher B wishes to study the association of TP53 polymorphism and lung-cancer. The objective is to find SNPs that may indicate a high risk for lung cancer. To do this study, B wants information from NCBI dbSNP about SNPs (chromosome location, SNP ID and build, gene location, codon, and protein), about alleles (amino acids and nucleotides), and about the nomenclature for amino acid levels and nucleotide levels. B also needs data about human subjects with lung cancer and needs to relate the SNP information to human-subject information.

To gather information from dbSNP, B constructs the form in the left panel in Figure 1. The form contains form fields for the data items B wishes to harvest. B next finds a first SNP page in dbSNP from which to begin harvesting information. B then fills in the form by cut-and-paste actions, copying data from the page in the center panel in Figure 1 to the form in the left panel.

To harvest information from the numerous other dbSNP pages, B gives the KBB a list of URLs, as the right panel in Figure 1 illustrates (although there would likely be hundreds rather than just the six shown in Figure 1). The KBB

[5] This took two domain experts more than one month just for one SNP for one disease.

Fig. 1. Form Filled in with Information from an SNP Page.

automatically harvests the desired information from the dbSNP pages referenced in the URL list. Since one of the challenges bio-researchers face is searching through the pages to determine which ones contain the desired information, the KBB provides a filtering mechanism. By adding constraints to form fields, bio-researchers can cause the KBB harvester to gather information only from pages that satisfy the constraints. B, for example, only wants coding SNP data with a significant heterogeneity (i.e., minor allele frequency $> 1\%$). Because of this filtering mechanism, B can direct the KBB to search through a list of all pages without having to first limit them to just those with relevant information.

For the research scenario, B may also wish to harvest information from other sites such as GeneCard. B can use the KBB with the same form to harvest from as many sites as desired. Interestingly, however, once the KBB harvests from one site, it can use the knowledge it has already gathered to do some of the initial cut-and-paste for B. In addition to just being a structured knowledge repository, the KB being produced also becomes an extraction ontology capable of recognizing items it has already seen. It can also recognize items it has not seen but are like items it has seen. The numeric data values or DNA snippets need not match precisely with those previously seen; they only need to be numeric values in a proper range or valid DNA snippets.

Using KBs as extraction ontologies also lets bio-researchers search the literature. Suppose B wishes to find papers related to the information harvested from the dbSNP pages. B can point the KBB to a repository of papers to search to cull out those that are relevant to the study. Using the KB as an extraction

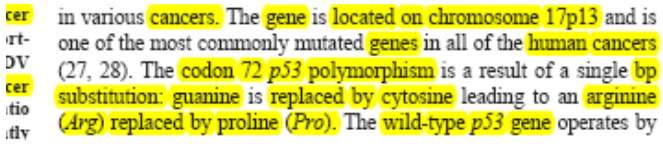

Fig. 2. Paper Retrieved from PMID Using a Generated Extraction Ontology.

ontology provides a sophisticated query of the type used in information retrieval resulting in high-precision document filtering. For example, the extraction ontology recognizes the highlighted words and phrases in the portion of the paper in Figure 2. With the high density of not only keywords but also data values and relationships all aligned with the ontological KB, the KBB designates this paper as being relevant for B's study.

For the human-subject information and to illustrate additional capabilities of the KBB, we suppose that a database exists that contains the needed human-subject information. The KBB can automatically reverse-engineer the database to a KB, and present B with a form representing the schema of the database. B can then modify the form, deleting fields not of interest and rearranging fields to suit the needs of the study. Further, B can add constraints to the fields so that the KBB only gathers data of interest from the database to place in its KB. Figure 3 shows an example of a form reverse-engineered from the Indivo database and altered to fit our research scenario.[6]

With all information harvested and organized into an ontology-based knowledge bundle (the KB), B can now do some interesting queries and analysis with the data. Figure 4, for example, shows a SPARQL query requesting the SNPs associated with any one of four amino acids: *Arg*, *Gly*, *Leu*, or *Trp*. For our example, we query based on information harvested from the six URLs listed in Figure 1. The query finds three SNPs and for each, returns the dbSNP ID, the gene location, and the protein residue it found. In our prototype, users may click on any of the displayed values to display the page from which the value was extracted and to highlight the value in the page. As Figure 4 shows, users may alternatively click on one or more checkboxes to access and highlight all the values in checked rows. The values *rs55819519*, *TP53*, and *His Arg* are all highlighted in the page in the right panel of Figure 4.

3 KBs and the KBB

Having provided a scenario in which a bio-researcher can use KBs built synergistically through a KBB, we now explain exactly what a KB is and how

[6] Since the Indivo schema has more tables and attributes than B wants, B selects only those tables and attributes relevant to the study before reverse engineering and tailoring the form. In many similar instances, preselecting before reverse engineering may be necessary to make the task of tailoring the resulting form reasonable.

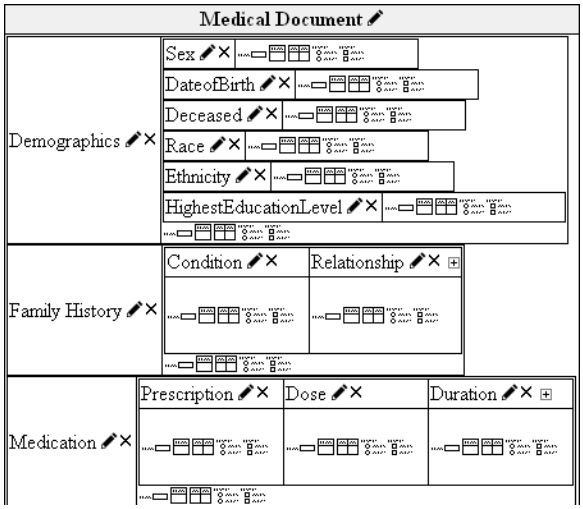

Fig. 3. Human Subject Information Reverse-Engineered from INDIVO (partial).

a KBB synergistically builds them. In doing so, we emphasize that although our research-study scenario specifically targets bio-research, our definitions and explanation here do not. Thus, KBs and KBB development of KBs can serve researchers, investigators, and decision makers in all disciplines.

We define a *knowledge bundle* (KB) as a 6-tuple (O, R, C, I, A, F).

- O is a set of object sets—sometimes called concepts or classes; they may also play the role of properties or attributes. (Examples: *Person*, *Amino Acid*, *Country*, *Color*.)
- R is a set of relationship sets among the object sets. (Examples: $Person(x)$ *is citizen of* $Country(y)$, $Sample(x)$ *taken on* $Date(y)$.)
- C is a set of constraints that constrain the objects and relationships in O and R. (Example: $\forall x(Sample(x) \Rightarrow \exists^1 y(Sample(x)$ *taken on* $Date(y)))$)
- I is a set of object and relationship instances embedded in O and R that satisfy C. (Examples: $Color($"green"$)$, $Sample($"SMP9671"$)$ *taken on* $Date($2009-03-25$)$.)
- A is a set of annotations for object instances in O; specifically, each object o in O may link to one or more appearances of o in one or more documents.
- F is a set of data frames [Emb80]. Data frames include recognizers for object and relationship instances as they appear in documents, instance converters to and from internal representations, operations over internal representations, and recognizers for operation instantiations as they appear in documents and free-form user queries.

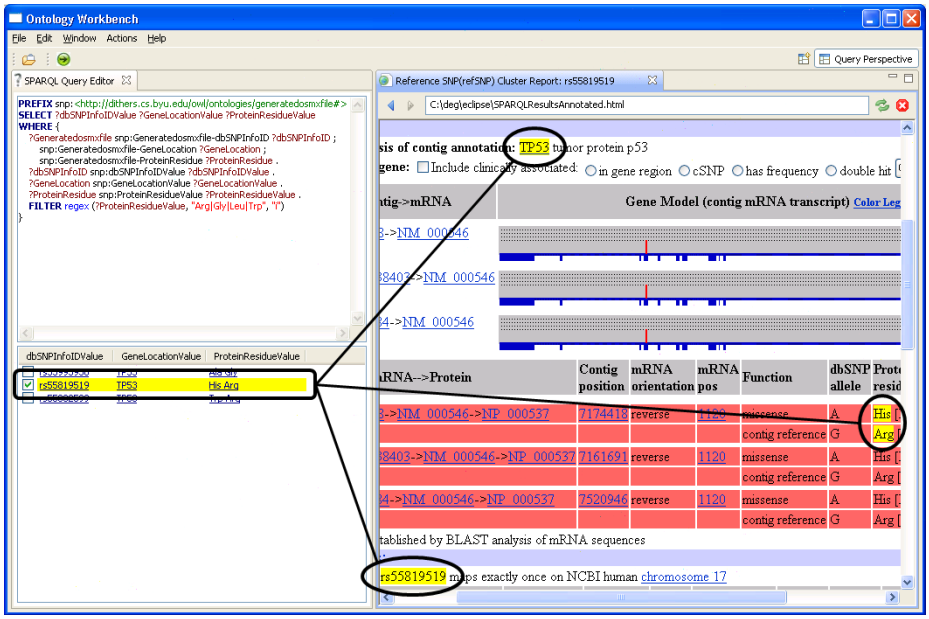

Fig. 4. Screenshot of our Current Web of Knowledge Prototype System.

The triple (O, R, C) is an ontology.[7] In our implementation, we use OWL to represent ontologies and RDF for storing instances with respect to OWL ontologies. Because a KB is a populated ontology, it is also a database and thus can be queried and mined for knowledge nuggets. Because a KB includes annotations, usually for each object instance, it provides a simple type of provenance—a link back to documents from which object instances are extracted. And because a KB includes data frames for object and relationship sets, it is an extraction ontology—an ontology that can recognize object and relationship instances in structured, semi-structured, and unstructured documents.

A *KB-Builder* (*KBB*) is a tool used to build KBs—more specifically, it is a tool to largely automate the building of KBs. The KBB has the capability to fully and automatically build a KB by reverse-engineering structured and semi-structured information sources into KBs. Perhaps more often than not, however, users need custom-built KBs. Using the KBB, a user U can start from scratch and build a KB from the ground up. U can specify a form and show how to fill in the form. The KBB watches and learns what U wants. Often, after only being shown how to harvest a handful of instances from machine-generated documents, the

[7] Researchers disagree about the definition of an ontology, but we adopt the view that an ontology is a formal theory and a specification of a shared conceptualization, both of which can be captured in a model-theoretic view of data within a formalized conceptual model. Since the elaboration of our triple (O, R, C) is a predicate-calculus-based formalized conceptual model [EKW92], we call it an ontology.

KBB can harvest and organize instances from hundreds of additional machine-generated, sibling, source documents. Further, as it collects more knowledge into its knowledge bundle, the KBB can create or identify domain-specific instance recognizers and use them to extract knowledge from as-yet-unseen, non-sibling, and even non-machine-generated source documents. Thus, the KBB can also do high-precision filtering to find additional relevant documents for the study. The KBB is synergistic, with as much of the burden shifted to the machine as possible. It allows users to check and fix any mistakes it makes, and it learns to do better as it is corrected. We now explain each major component of the KBB.

Form-based Ontology Creation. While we do not assume that bio-researchers and other decision-making researchers are expert users of ontology languages, we do assume that they can create ordinary forms for information gathering. The KBB interface lets users create forms by adding various form elements. Clickable icons in the data and label fields of the forms in Figures 1 and 3 let users control form creation. Users can specify any and all concepts needed for a study, can specify relationships and constraints among the concepts, and can nest, customize, and organize their data any way they wish. From a form specification, the KBB generates a formal ontological structure, (O, R, C). Each label in a form becomes a concept of O. The form layout determines the relationship sets in R among the concepts and determines the constraints in C over the concepts and relationship sets.

Information Harvesting. How well the KBB harvests information from a particular site depends on how regular the pages are. Most pages are uniform enough that the KBB can harvest information without user intervention.[8] For each item to be harvested from an HTML page, the KBB generates an xpath to the node in the DOM tree in which the item is located. For data items within the node, the KBB automatically infers the left and right context information it needs as well as delimiter information for list patterns. When the pages are not as uniform as might be expected, the KBB works interactively with a user U, allowing U to cut and paste any data items missed or mistakenly entered in its automated harvesting mode. The KBB learns from these corrections and makes adjustments as it continues to harvest from additional pages in the site. While harvesting information, the KBB builds the I and A components of a KB. Since the KBB harvests concept value instances and relationship instances with respect to the defined ontology, it is able to immediately populate the ontology with these harvested values and thus build the I component of the KB. Constructing the A component is a matter of keeping links to the pages and locations within the pages from which the value instances are extracted. The KBB records the xpath to the node in which the value appears and the offset within the node.[9]

[8] In preliminary evaluations we have conducted, the system has often been able to achieve 100% precision and recall [Tao08].

[9] Because web pages can change, we cache pages when we harvest them. This ensures that provenance links remain valid. It also means, however, that the KBB may need to reharvest information from the page when its content changes.

Extraction-Ontology Creation. Building the F component of a KB turns the populated ontology into an extraction ontology. Ontologies augmented with instance recognizers are called *extraction ontologies*. Instance recognizers, contained in *data frames*, are regular expressions that recognize common textual items such as dates, email addresses, and DNA sequences. They can also contain lexicons that match with items such as car makes and models and company names.[10] To build an extraction ontology and thus the F component of a KB, the KBB needs to be able to create instance recognizers. The KBB creates instance recognizers in two ways as it harvests information: (1) by creating lexicons and (2) by identifying and specializing data frames in a data-frame library. Lexicons are lists of identifiable entities—e.g., lists of country names. As the KBB harvests country names or any other named entity, it stores the unique names in a lexicon. Thus, when the KBB encounters the name again, it can recognize and classify it. For data-frame recognizers, we initialize a data-frame library with data frames for common items we expect to encounter—e.g., all types of numbers, currencies, postal codes, and telephone numbers, among many others. When recognizers in these data frames recognize harvested items, they can classify the items with respect to these data frames and associate the data frames with concepts in the ontology. Some automatic specializations are possible, such as numbers with as-yet-unrecognized units. For more complex pattern recognition, experts can add domain-specific recognizers as needed.

Reverse-Engineering Structured Data to KBB Forms. Structured repositories (e.g., relational databases, OWL/RDF triple stores, XML document repositories, HTML tables) may contain much of the information needed for a research study. A reverse-engineering process can turn these structured repositories into knowledge bundles. Figure 3 shows an example of a generated KB form resulting from a reverse-engineering process. Researchers can custom-tailor reverse-engineered KBs by restructuring the generated forms to become the (O, R, C)-ontologies they want. They can also limit the data extracted from the database to the I-values they want, and they can employ the techniques mentioned in the previous paragraph to produce F-component lexicons and data frames for extraction ontologies from data in the database.

KB Usage for Analysis and Decision Making. As the KBB harvests information, it stores harvested information in its KB, which is a knowledge base.[11] Users can query and mine the KB in standard ways. Query results, however, have an additional feature beyond standard query results—each result data value v is "clickable" yielding the page with v highlighted as Figure 4 shows.

[10] Much has been said about extraction ontologies, including experimental results that generally show that for many applications, precision and recall is high—over 90% for many concepts and data values (e.g., [ECJ+99]).

[11] Note that KB, the acronym for a knowledge bundle, is the same as the acronym for a knowledge base. This is intentional—a knowledge bundle includes a knowledge base, usually personalized for some specific research agenda.

4 Implementation Status and Future Work

We have implemented an initial prototype of our KBB as part of our web-of-knowledge project [ELL+08]. Currently, as Figure 1 shows, users can create ontologies via our forms interface, fill in the form from a machine-generated web page from a hidden-web site, and harvest information from the remaining sibling pages of the hidden-web site [Tao08]. Also, as Figure 3 illustrates, we can reverse-engineer machine-generated sibling tables from hidden-web sites into forms, and we can automatically establish the beginnings of an extraction ontology embedded within a KB [Tao08]. Using extraction ontologies coded by hand, we have successfully been able to do high-precision filtering of semi-structured web documents [XE08], but we have not yet brought this up to the level we need for high-precision document retrieval for free-running text as indicated in Figure 2. Our current implementation also allows users to access and query the data in a KB as the screenshot of our working prototype in Figure 4 shows.

Although some of our work is complete, we still have much to do to solidify and enhance what we have already implemented and to extend it to be a viable research-study tool. We plan to further our research as follows.

- *Relationships.* We have defined and implemented data frames for concepts corresponding to nouns and adjectives. To accommodate relationships explicitly, rather than implicitly as we do now, we plan to define data frames for relationships in connection with verbs and prepositions. We may also need to adapt techniques from natural language processing and probabilistic grammars.
- *Boundary relaxation for free-running text.* Our current system expects source documents divided into distinct records. This has been useful in dealing with tables and computer-generated output. However, in order to extract selected information from free-running text, we need to relax the boundary constraints, and eventually to be able to recognize a record of interest, and its extent, without any boundary information.
- *Reverse engineering of structured data.* Our reverse-engineering efforts have proven to be successful for XML [AKEL08] and machine-generated sibling tables [Tao08], and we have done some reverse-engineering work for relational databases and OWL ontologies. We should, however, take these approaches even further, for instance, by inferring schemas from general semi-structured data, and then reverse-engineering those schemas to KBs. Specifically, we would like to be able to reverse-engineer machine-generated web pages with a mixture of tables, lists, and and other semi-structured information such as the dbSNP page in Figure 1.
- *Specialized information gathering.* Besides adding constraints to limit the potentially overwhelming amount information to be harvested, the KBB should support more sophisticated kinds of specialized information gathering. Can we add features to KBs to do hypothesis testing? Extending KBs with hypothesis declarations and rules and reasoning capabilities could give it the ability to locate and extract evidence to substantiate or refute hypotheses.

– *Quality assurance.* How do we guarantee that our KB is correct? As currently implemented, our only guarantee is a provenance link to the original pages from which information is taken. Issues of integration, record linkage, data cleaning, and data fusion each play a role in quality assurance. Having had some experience with these issues (e.g., [XE06]), we know that the many problems are varied and tough. As KBs come to be relied upon as knowledge sources, however, we must find ways to guarantee their quality.

5 Concluding Remarks

Several related fields of research are at the heart of our work: information extraction [Sar08], information integration [ES07], ontology learning [Cim06], and database reverse engineering [MH08].[12] Our contribution is in drawing from each of these fields, mostly from our own research work, those components that together establish the basis for KBs and KBBs. The KB/KBB approach discussed here is a unique, synergistic blend of techniques resulting in a tool to efficiently locate, extract, and organize information for research studies. (1) It supports directed, custom harvesting of high-precision technical information. (2) Its semi-automatic mode of operation largely shifts the burden for information harvesting to the machine. (3) Its synergistic mode of operation allows research users to do their work without intrusive overhead. The KB/KBB tool is a helpful assistant that "learns as it goes" and "improves with experience."

We note that KBs are not for everyone's information-finding needs. Although semi-automatic with much of the burden being shouldered by the KBB, the overhead in establishing KBs from scratch may outweigh the advantages gained. Individuals just wanting a quick answer to a question would not, indeed should not, use the KBB to build a KB from scratch to answer their question. On the other hand, it is not hard to envision a web of thousands of KBs already built, interlinked on identical data items, and publicly available. In this case, the web of KBs would directly support question answering—returning answers to questions and links to pages to verify answers. Indeed, as explained elsewhere [AME07], users can successfully pose free-form questions in an interface to a web of KBs. We further note that when building a KB is appropriate, none of the overhead is wasted. Users simply start harvesting and organizing the information they need under the "watchful eye" of the KBB, which takes on ever more of the task.

References

[AKEL08] R. Al-Kamha, D.W. Embley, and S.W. Liddle. Foundational data modeling and schema transformations for XML data engineering. In *Proceedings of the 2nd International United Information Systems Conferences (UNISCON'08)*, pages 25–36, Klagenfurt, Austria, April 2008.

[12] Each of these fields, by itself, is huge. Of the hundreds of papers we could cite, we thus only cite a recent survey or book for each.

[AME07] M. Al-Mumammed and D.W. Embley. Ontology-based constraint recognition for free-form service requests. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE'07)*, pages 366–375, Istanbul, Turkey, April 2007.

[Cim06] P. Cimiano. *Ontology Learning and Population from Text: Algorithm, Evaluation and Applications*. Springer Verlag, New York, New York, 2006.

[ECJ+99] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.

[EKW92] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-oriented Systems Analysis: A Model-Driven Approach*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[ELL+08] D.W. Embley, S.W. Liddle, D.W. Lonsdale, G. Nagy, Y. Tijerino, R. Clawson, J. Crabtree, Y. Ding, P. Jha, Z. Lian, S. Lynn, R.K. Padmanabhan, J. Peters, C. Tao, R. Watts, C. Woodbury, and A. Zitzelberger. A conceptual-model-based computational alembic for a web of knowledge. In *Proceedings of the 27th International Conference on Conceptual Modeling*, pages 532–533, Barcelona, Spain, October 2008.

[Emb80] D.W. Embley. Programming with data frames for everyday data items. In *Proceedings of the 1980 National Computer Conference*, pages 301–305, Anaheim, California, May 1980.

[ES07] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, Heidelberg, Germany, 2007.

[MH08] N.A. Mian and T. Hussain. Database reverse engineering tools. In *Proceedings of the 7th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, pages 206–211, Cambridge, United Kingdom, February 2008.

[Sar08] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[Tao08] C. Tao. *Ontology Generation, Information Harvesting and Semantic Annotation for Machine-Generated Web Pages*. PhD dissertation, Brigham Young University, Department of Computer Science, December 2008.

[XE06] L. Xu and D.W. Embley. A composite approach to automating direct and indirect schema mappings. *Information Systems*, 31(8):697–732, December 2006.

[XE08] L. Xu and D.W. Embley. Categorization of web documents using extraction ontologies. *International Journal of Metadata, Semantics and Ontologies*, 3(1):3–20, 2008.