Exploring the Deep Web: Associativity Search over Schematic Metadata

Govind Kabra[1], Zhen Zhang[1], Kevin Chen-Chuan Chang[1], Lipyeow Lim[2], Min Wang[2], Yuan-chi Chang[2]
[1]University of Illinois at Urbana-Champaign, [2]IBM T.J. Watson Research Center

ABSTRACT

The Web has been rapidly deepened with the prevalence of databases online. As sources proliferate, while there are often useful, alternative, and related sources for our needs, we are lacking an effective facility to explore this "deep Web." For "ad-hoc users" and "system integrators" alike, to enable access and integration to the multitude of sources, we often must answer semantic association questions– How sources relate to each other? What "vocabularies" do they speak? Such semantic associativity is often revealed holistically through cooccurrence analysis of "schematic metadata," which describes the nature of data at sources. We observe two interesting phenomena through the syntactic associativity of sources and their schematic metadata: The first phenomenon, *occurrence localities*, suggests syntactic associativity as a useful notion for discovering semantic associativity, and the second, *fuzzy boundaries*, suggests a query-driven rank-based mechanism as its realization. We thus propose to build an *associativity search* facility for systematic exploration of deep Web sources. In its realization, we combine occurrence analysis and link analysis by abstracting occurrence of metadata in sources as links in a graph, which effectively transforms associativity of entities into connectivity of nodes. To quantify the associativity, we propose a wave propagation model; to compute the associativity efficiently, we develop spatial and temporal optimization strategies. We validate the usefulness and efficiency with a real-world dataset of 30,000 sources. The experiments show that syntactic associativity is not only useful for semantic discovery, but also practical as an online search mechanism.

1. INTRODUCTION

With the increasing presence of databases online, the Web has been rapidly "deepened"; many sites now provide dynamic content through query interfaces (instead of static URL links). The recent surveys [3, 4] estimated 10^5 such databases, in a wide range of diverse domains. With this large and ever expanding scale, the proliferation of this "deep Web" has created an interesting challenge: How to explore the abundance of data sources in this open Web? Much like current search engines guide navigation on the "surface" Web, such exploration facilities will open up our access to and integration of the deep Web.

To begin with, the deep Web, with its massive data sources, provides unprecedented opportunities for *information access* in various domains, often with abundance of alternative and related sources. The wide range of sources can meet diverse needs of everyday users. For instance, for buying books, an ad-hoc user Amy may search *amazon.com* or, alternatively, *bn.com*. Then, for travel planning, she may visit *aa.com* as well as the related *hertz.com* and *hotels.com*. On the open network of the Web, information access starts with finding "*where and what sources are?*" in terms of how their data match our needs and how they relate to each other.

Further, with many alternative and related sources, it also presents immense demands for *information integration* over such "companion" sources. Such integration will build vertical search engines, metasearch systems, and domain-based portals for combined and comparative accesses to multiple sources of specialized needs. To integrate autonomous sources is, in essence, to "speak" their query languages. To realize integration, a system integrator Sue will need to know, while building a unified query interface for, say, "books," that attributes author, title, and isbn are common to such sources. Likewise, she may want to find sources that can accept make and model queries, *e.g.*, *cars.com* and *edmunds.com*. Thus, to bring together sources online, information integration requires knowing "*what 'vocabulary' sources commonly speak?*" in terms of representative and related attributes (among other query properties).

We thus believe, for ad-hoc users and system integrators alike, the ability to explore sources, in terms of what they are about and what their vocabularies are, is imperative for opening access to the deep Web. As Amy may ask, what sources are related to *aa.com*? As Sue may ask, what do sources like *amazon.com* and *bn.com* accept for querying? In many similar scenarios, we are looking for *semantic associativity* between sources, between their vocabularies, and between sources and their vocabularies. Lacking such facilities, the deep Web remains an uncharted territory, with unrealized promise for information access and integration.

This paper aims at building a unified, systematic facility for answering various such semantic associativity questions. As *source models*, how to describe each source? As *search mechanisms*, what functions should we provide? We address these issues in turn.

To start with, we propose a simple source model, which captures a source by its *schematic metadata– e.g.*, query attributes (author, isbn); representative keywords ("books", "fiction"). Such modeling is an interesting practical issue. As data are hidden behind query forms, how to capture the nature of a source without "crawling" deep into its content, which can be expensive, requires query wrapping, and may drain a source. Our simple model thus uses only schematic features, which reflect functionally the types of the underlying data (*e.g.*, isbn is implied from books as data). While not our focus here, we note that such metadata features are extractable from the "surface" with current techniques (*e.g.*, by attribute extraction [26] and keywords selection [25]).

Upon such a metadata repository with schematic models of online sources, we propose to search explicit *syntactic associativity* as an algorithmic mechanism for exploring implicit semantic associations. The observations from syntactic association (as occurrence and cooccurrence) to semantics relevance have been explored in information retrieval in various forms (*e.g.*, the vector space model uses "first-order" cooccurrence, and query expansion [24]

second-order). As our foundation, such phenomena are further reinforced for schematic metadata of Web sources: Several recent efforts [13, 23] employ "holistic schema matching," by analyzing cooccurrences of attributes across many sources, to discover their semantic matching. The effectiveness results from, *first*, that schematic metadata (unlike "noisy" words in documents) functionally reflects the semantic nature of source data. *Second*, as Section 3 will observe, Web sources seem to exhibit some interesting "Amazon effect," where peer influences lead to more regular and converging vocabularies with clear cooccurrence "localities."

We thus aim at building a systematic cooccurrence-analysis facility, as an online *associativity search* mechanism, for exploring the metadata repository. While cooccurrence has been studied in various contexts, for isolated and offline tasks, we attempt to distill such efforts into a unified mechanism for searching various syntactic associativity. To support ad-hoc users and system integrators with diverse needs, and to find associativity of various "strengths," our search mechanism will be *query-driven* and *rank-based– e.g.*, given query $Q = \{$*amazon.com*, *bn.com*$\}$ to find associated sources (*e.g.*, *borders.com*, *powells.com*, $\cdots$) and their metadata properties (*e.g.*, title, author, isbn, $\cdots$), in ranked orders.

Our approach essentially hinges upon a novel combination of *cooccurrence* analysis with *link* analysis. We view our metadata repository as a *graph*, where each occurrence is a *link* between a metadata property and a source– *i.e.*, a metadata graph with a topology matrix W capturing occurrences between nodes. We thus take a conceptual view that $\mathcal{A}(t|Q)$, the associativity of a node t to query $Q = \{a_1, \cdots, a_n\}$, is how we can "reach" from $a_1, \cdots, a_n$ to t through, transitively, all paths and multiple hops of occurrence links. To quantify such "connectivity", we assume a "ripple wave" simulation [8], where multiple waves are originated at $a_1, \cdots, a_n$, propagate through links with decays, and combine into a steady-state wave of certain "magnitude" at target node t– We interpret this magnitude as $\mathcal{A}(t|Q)$. The simulation amounts to a Markov chain propagation on W, with the solution as its eigenvector. This framework thus transforms cooccurrence analysis into principled link analysis, to search from any nodes their companion nodes with quantified associativity measure. Aiming such service as an interactive search mechanism, we further develop "spatial" decomposition and "temporal" aggregation as optimizations to speed up the computation.

We have implemented this associativity search facility, and experimented it over our repository of about 30,000 sources crawled from the Web. The results show that 1) syntactic associativity is useful for answering various types of semantic associativity questions, 2) it can be efficiently supported as an online search mechanism, with sub-seconds response time. While this paper focuses on the deep Web, we believe such metadata exploration facilities will be crucial for any open environment with dynamic sources (a clear departure from the "closed" world of RDBMS catalog registries), such as the emerging notion of open "dataspace" [2] and schema corpus [12]. In summary, this paper makes the following contributions:

- **The Problem of Metadata-based Source Exploration**: We identify the problem of exploring data sources proliferating online, to facilitate information access and integration.
- **The Framework of Associativity Search**: As our solution, we develop associativity search as a novel transformation from cooccurrence analysis to link analysis, thus enabling a systematic and unified framework for searching associativity.
- **The Mechanism of Online Computation**: We present spatial and temporal optimization techniques to speed up matrix computation, with satisfactory performance for interactive search.

With Section 2 for the related work, we start in Section 3 to motivate the problem and our abstraction. Section 4 presents the framework for quantifying associativity, and Section 5 the computation mechanism. Finally, Section 6 presents our experiments.

2. RELATED WORK

Database sources on the deep Web have gained much attention recently. Several problems have been studied, such as, crawling [21, 14], interface understanding [26], schema matching [13, 23, 22, 12] and query translation.

Associativity of attributes have been useful in building clustering-based [23] as well as correlation mining based [13] schema matching techniques. An extensive survey [4] on deep Web reports that the attribute of related sources on deep web exhibit strong localities. Inspired by these evidences, we propose a framework for exploring the associativity of such metadata.

Exploiting a large repository of schemas with associated data and metadata has been proposed earlier [12], however, this work focuses on its application to schema matching. Ours is the first attempt towards developing a systematic framework for exploring the repository of deep web sources.

Several efforts towards integration on deep web [26, 15, 23, 22] focus on "domain-based" integration. These solutions can use our framework to obtain sources in specific domain.

The two core aspects of our associativity based exploration framework are: (a) An *occurrence based associativity* model, and (b) A mechanism for relevance *propagation across link structure* formed by such associations.

Occurrence based associativity: Several other works also use the occurences or cooccurrences to capture the semantic associativity. Association rule mining [1] uses occurrences of items in transactions. However, in contrast to our approach, it does not allow transitivity over these occurrences, but rather only uses the first-order occurrence. Further, it mines associations independent of user query.

Query Expansion [24] also uses term-document occurrence analysis to find related terms for expanding queries using a local or a global analysis. While both the approaches have shown only a limited success in IR, these techniques can possibly support the functionality $T1$ in our model. However, they will require, as input, the number of iterations of query expansion, number of documents to consider (for local analysis), number of terms to choose for expansion, etc. Our framework does not require such fine tuning, supports all the four functionalities $T1$ - $T4$, and can model not only keywords but multiple schematic metadata.

Latent Semantic Indexing or LSI [7] uses singular value decomposition for reduction of original occurrence-based matrix to k dimensions. Similar to our top-k spatial optimization, this also requires as input the value of k, which happens to be a major reason for limited success of both the techniques.

Propagation across link structure: Link analysis has been used to compute the PageRank [20]of web pages. Since our problem is query-driven, it more related to Personalized PageRank [17]. In comparision to PageRank, our problem has several differences: (a) (Personalized) PageRank only forms a part of the overall scoring function, and so it has to be computed for every node in the graph. In contrast, in our problem the score obtained using propagation are the final score, and so it is sufficient to compute scores only for the more relevant nodes. (b) Pagerank is computed in an offline fashion. Our problem requires interactive response time. Personalized pagerank is also required to be computed for every query, however,

the model is still far from realization. (c) PageRank computation has billions of pages, while we expect the scale of our problem to be in millions of sources.

Spreading Activation or SA techniques [6] use sophisticated algorithms for relevance propagation using the constraints based on the semantic information in links of inference or neural network. In contrast, we derive links purely using "syntactic" occurrence analysis, and mathematically model them in a linear system with Markov chain.

3. MOTIVATION: ASSOCIATIVITY SEARCH

As structured data sources are proliferating on the Web, for a particular type of information (*e.g.*, finding flights) there are often "alternatives" (*e.g.*, *aa.com* and *united.com*) or "related" (*e.g.*, *hertz.com*, *hotels.com*) sources, which we call generally *companion* sources. As our information access often needs to consult multiple related sources, the *access* to the deep Web often starts with finding such sources for specific needs. Further, with many sources amass online, *integration* is, by definition, to explore and bring together such companion sources. The deep Web, with a multitude of sources online, provides novel opportunities for accessing and integrating comparative or complementary information.

3.1 The Needs: Semantic Associativity

While there exist many useful sources, with the lack of an effective exploration facility, the deep Web remains an uncharted frontier for information access and integration. Such lacking is pervasive in various scenarios, for not only *ad-hoc users* who come with different "everyday" information requests (finding flights, houses, or jobs) but also *system integrators* who aim at building integration systems such as comparison shopping, vertical search engines, or specialized portals across sources in one or several related domains (*e.g.*, travel, real estate, or jobs). To begin with, when users are finding sources to access, or when integrators are exploring sources to integrate, they will ask some common questions, centering around searching for sources by certain "clues":

- $T1$: What are sources that are alternative to *amazon.com*? Related to *aa.com*?
- $T2$: What are sources about "books"? Or "flights"? What are sources that can answer queries like "make = GM"? Or, queries about author and title?

Further, for bringing together sources in the same or related domains, integrators often need to know about their query "vocabulary." They may ask such questions:

- $T3$: What are common queries used by *amazon.com* and *a1books.com*? What are common keywords used by these sources?
- $T4$: What are common attributes used with title and author? What are queries supported by sources about "books, fictions"?

To open up access and integration to various domains of online sources, it is crucial to answer these questions– to know where sources are, what (keywords) they are about, and what (queries) they speak– which are essentially "semantic associativity" between sources and their "vocabulary" (*e.g.*, common keywords and query attributes). Such semantic associativity is established through the underlying types of data provided by sources. Sources are semantically associated if they provide the same or similar types of data objects (*e.g.*, both *amazon.com* and *a1books.com* for books, and *aa.com* and *united.com* for flights). All above questions can be cast as a form of such semantic associations– *e.g.*, $T1$: How a source associates with other sources? $T2$: What sources associate with keyword "books"? Such semantic associativity, suggesting (albeit subjectively) relevant sources, is crucial for guiding our access to useful online sources. Further, by exploring how sources relate to each other, such an associativity is also important for integration, which aims exactly at bringing companion sources together.

We note that, while useful, such queries are *fuzzy* in nature, and their answers often depend on application needs: For instance, for $T1$, while *united.com* is related to *aa.com* as an alternative site, perhaps so are *hertz.com* and *hotels.com* as complementary ones. Similarly, for $T2$, probably all these sites are related to keyword "travel" or query "arrival = 03/16/2006". Such fuzziness is natural, because sources and their vocabularies are related in multiple ways– *e.g.*, *amazon.com* is a "book" source and a "music" source (among others); *hertz.com* is not only "car rental" but also "travel"'. As the answers are not "black and white", these questions demand flexible answers (e.g., ranking in a way similar to what Web search engines do) that will accommodate different application needs.

While these questions are important and pervasive, we are clearly lacking a unified, systematic facility to explore data sources on the deep Web– Today's search facility on the Web, as their unintended side effects, can answer such queries only partially and in a tedious way: To begin with, with current search engines, one may find *similar pages* to `http://www.aa.com`– finding such similar pages (and not necessarily data sources) may return potential data sources (and other pages), but how are results ranked? Is the "page-oriented" semantics appropriate for relating data sources? Can I search by query capabilities? While there are several search systems that have been engineered specifically for "searching" deep Web sources, *e.g.*, *completeplanet.com*, *www.Goshme.com* or *invisible.net*, they are largely based on similar concepts of keywords search– that of matching sources by keywords– which would address $T2$ partially and only.

As data sources continue to amass on the Web, we aim at building an effective online "mining" facility for Web sources, in analogy of today's search engines for pages. Upon a *repository* of sources collected from online, such a mining facility will model each source with its key characteristics, index such parameters, and provide an efficient search function to help answer questions like $T1$ to $T4$ over the set of sources. This paper focuses on supporting such exploration facilities over a source repository, and not their collection issues. By crawling techniques (as we reported in [5]), which recognize a data source by the existence of a query interface as its "entrance" to the underlying database, we can construct such a repository of various scales, depending on the application focus– say, comprehensively for any sources, or only for those related to certain subjects, *e.g.*, "travel" or "housing." Such repositories already exist today: *completeplanet* claims 70,000 sources, *invisibleweb.net* 4,000; our experimental dataset (Section 6) consists of 30,000 with our own crawling. While such source repositories already exist, however, the lacking of an exploration facility amounts to two challenges:

Lack of Source Models: For building such a facility, as the foundation, we are lacking an information model for describing each source. To answer such "semantic" questions as $T1$, $T2$, $T3$, and $T4$, what do we need to know about each source? How do we capture such a "model" and to abstract the exploration facility as a search problem?

Lack of Mechanisms: What machinery, then, will provide the necessary functions useful for semantic-associativity questions like $T1$ to $T4$? As users interests are ad hoc and diverse, we need to provide a flexible and interactive facility for exploring the source repository.

3.2 Opportunities: Syntactic Associativity

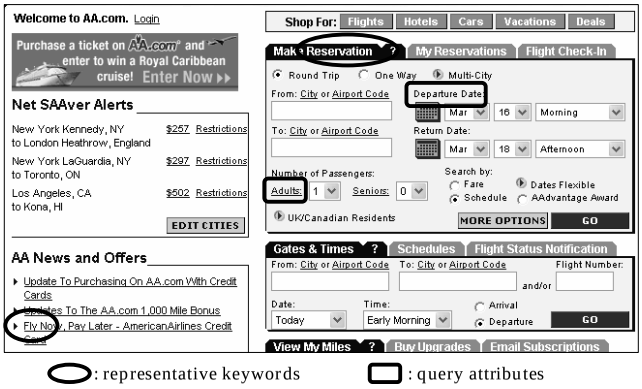


Figure 1: Query interface page: *aa.com*.

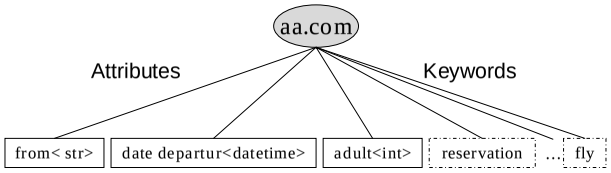


Figure 2: Schematic metadata of *aa.com*

Source Model: Schematic Metadata. The very first challenge in building an exploration facility is to develop a source model, upon which semantic associativity can be discovered. As exploration is the preamble to information access and integration, our source model must be easily obtainable through "surface" without "entering" a source, which, if so, would require more sophisticated understanding of the source. As entrance to the underlying data of a source, a query interface, often provides rich "metadata" information describing what the source is about and what they speak. For instance, from the interface page of *aa.com*, as Figure 1 shows, we can easily identify some *representative keywords* such as "reservation," "fly" describing the functionality of the source, and *query attributes* such as "departure date," "adults" describing the schema of the data. Such metadata provides high-level abstraction of a source, and thus gives us good hints on semantic associativity.

To establish the semantic associativity upon the underlying types of data objects, our metadata must be functionally determined by this type of data provided by a source. We therefore generally refer to such metadata as *schematic metadata*. As the above example illustrated, for a source providing a particular type of information, *e.g.*, airfares, the representative keywords (*e.g.*, "reservation") and queriable attributes (*e.g.*, departure date) are generally determined by the type of service provided. Some other types of metadata, such as size of the databases or PageRank of the source, are generic to sources of different types, and thus not helpful for relating companion sources upon semantic associativity.

The schematic metadata gives us an effective mechanism to characterize and relate sources. Since schematic metadata describes the nature of the data, sources that share similar schematic metadata therefore provide similar types of data, and thus is semantically associated. For instance, airline sources may relate by typical keywords such as "flight, reservation, tickets," or attributes "departure city, departure data" *etc.*. Specifically, we define our schematic metadata model as follows:

To relate companion sources, we model the source with schematic metadata, or metadata for short. Specifically, the metadata of a source is a container, containing a set of properties, describing the type of data provided in the source from various aspects.

Definition 1 (Schematic Metadata Model): The metadata of source s consists in m types of *property entities* $\{s.P_1, \ldots, s.P_m\}$. For each type $i, i = 1, \ldots, m$, $s.P_i$ is a set of *property entities*, describing the source from a particular perspective. ■

	output	
input	sources	metadata
sources	$T1$	$T3$
metadata	$T2$	$T4$

Figure 3: Semantic associativity: sources and metadata.

Specifically, in modeling the deep Web sources, we find two types of properties are of particular importance: attributes P_a to specify the schema of a source, and typical keywords P_k, or keywords for short, to characterize the function of a source. As motivated in question $T1$ to $T4$, the exploration of sources often involve these two aspects. Therefore, in this paper, we focus the metadata model on these two types. Specifically, like a column in a relation, an attribute $name[type]$ specifies the name of the attribute $name$ and its data type $type$ (*e.g.*, string, int or float). Based on this model, Figure 2 shows the metadata of *aa.com*. For instance, attribute from is a property entity of str type, and adult int type. "Fly" is a keyword property of *aa.com*. We will discuss how to derive the metadata of a source in Section 6.

Such schematic metadata, as our source model, not only provides the basis for establishing semantic associativity, but also plays a central role, both as input and output, in various semantic questions. In fact, the four questions $T1$ to $T4$ as Section 3.1 discussed can all be cast as instances of finding semantic associativity between {source, metadata} $\rightleftharpoons$ {source, metadata}, as Figure 3 shows. Specifically, question $T1$ asks for semantic associativity from source $\rightarrow$ source, $T2$ metadata $\rightarrow$ source, $T3$ source $\rightarrow$ metadata and $T4$ metadata $\rightarrow$ metadata.

Mechanism: Syntactic Associativity. While the large scale of online sources imposes many challenging questions for their underlying "semantic associativity" (that cannot be directly observed), it also offers good opportunities to find interesting phenomena of "syntactic associativity"–*i.e.*, the associativity that can be revealed from syntactic features observable at sources. In terms of our schematic metadata for sources, we can observe their *occurrence*: how metadata is used by sources, and *containment*: vice versa. The associativity can immediately generate in transitions, to observe *co-occurrence*: how metadata is used together, or *co-containment*: how sources use the same metadata.

In fact, for our goal of exploring semantic associativity (*e.g.*, $T1$ to $T4$), we are inspired by several related, but specialized, efforts that exploit such syntactic cooccurrences for semantics discovery. For instance, in the related IR field for unstructured text, latent semantic indexing (*e.g.*, [7]) and query expansion (*e.g.*, [24]), all exploit the term cooccurrences to discover the "hidden" semantics for relate documents. While such syntactic associativity is not particularly effective in IR, its promises seem evident for our schematic metadata, which can be witnessed from several recent works with closer objective to ours. In the emerging deep Web integration efforts, for schema matching (to discover semantic correspondence between attributes between sources), people [13, 23] have pursued "holistic matching"– to perform correlation mining or clustering upon cooccurrences of query attributes across many sources.

To more systematically visualize such syntactic associativity and its strong indication of semantic associativity, we surveyed 500 sources [4] in 8 representative domains from the UIUC Web Integration repository. In this survey, we examined how attributes, as a particular type of schematic metadata properties, occur in sources and whether such syntactic occurrences reveal certain semantic as-

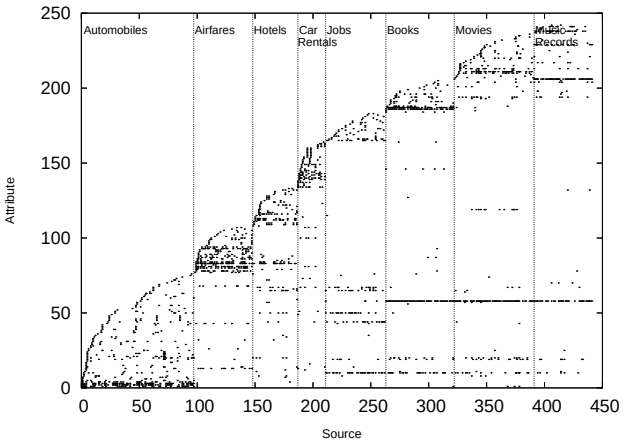


Figure 4: Attribute occurrences over sources.

sociativity. Figure 4 shows the survey result, where x-axis represents sources and y-axis attributes. A dot (x,y) in the Figure means attribute x occurs in source y. From the figure, we can clearly observe the densely populated triangle along the the diagonal line. This observation reveals the following phenomena:

- *Occurrence localities:* We observe a strong indication of "localities" of occurrences. The dots form triangles along the diagonal line. Each triangle suggests a locality resulting from the phenomenon that a small set of attributes often *cooccur* in a set of "similar" sources. Such localities nicely correspond to the "nature" of data domain of sources, *i.e.*, attributes (*e.g.*, author, isbn for books; make, model for cars) describing data of the same or similar domain cooccur in the same localities of sources, and vice versa, such sources (*e.g.*, *amazon.com*, *bn.com*; *cars.com*, *edmunds.com*) will cocontain localities of attributes.
- *Fuzzy boundaries, overlapping granularity:* While localities are evident, they are not cleanly separated: First, the boundaries can be blurred and interlinked: Although most dots appear in the diagonal triangles, there are many sparsely distributed "outliers," which indicates some attributes appear and interlink multiple localities. Further, as some of these "outliers" appear with more concentration across several neighboring domains and thus interlink them into localities of larger granularity, which exists between Books, Movies, and Music Records as well as between Airfares, Hotels, and Car Rentals. Figure 4 orders sources to make this "interlinking" more explicit.

Why do we observe such localities, which seemingly correspond to semantic associativity of sources (and their metadata)? Such observations are not surprising. As just mentioned, several recent efforts have exploited similar phenomena that databases on the Web are *not* arbitrarily complex– there seem to be some "convergence" and "regularity" *naturally* emerging across many sources. In hindsight, we believe, such behavior is indeed natural at a large scale, when data sources amass on the Web as "companions." As sources proliferate, they tend to be influenced by peers– which we intuitively understand as the *Amazon effect*[1]– for not only *what* their peers are doing but also *how* they do it:

- *What peers are doing* will influence proliferation of sources of similar natures. In analogy, as Amazon.com becomes a successful online bookstore, many other sources will emerge to serve the same function, or the same type of books data. Thus, we expect sources of similar natures will proliferate. Thus, the locality of sources (serving similar data) will naturally emerge, within the same domain (*e.g.*, Books) or more generally in related domains (*e.g.*, Movies, Music Records).
- *How peers are doing* will influence the "model" of companion sources. In analogy, as Amazon.com becomes a popular online bookstore, other book sources, when serving the same type of data, will likely follow its "de facto" convention of how books are described and queried. Thus, the locality of attributes, or schematic metadata in general, will naturally emerge, as sources of the same nature tend to use similar vocabularies.

[1]Online bookstores seem to follow Amazon.com as a de facto standard.

Together, it seems intuitive and natural that, for serving certain data (especially those with popular demands), similar or related sources will amass online, and they will share common vocabularies. Therefore, as we model sources by adopting their representative vocabularies (keywords, query attributes) as source schematic metadata, sources thus will form localities by co-containing common metadata properties, and vice versa. metadata properties will form localities by co-occurring in companion sources.

Such *occurrence locality* phenomena, which we will generally refer to by *syntactic associativity* between sources and their schematic metadata, are indicative of their semantic associativity. As Figure 3 shows, our tasks are, in essence, to systematically associate sources and their metadata in all possible ways. Our observations suggest that the key to discovering semantic associativity is to explore syntactic associativity by occurrence analysis, to output sources or metadata relating to input ones in terms of how they cooccur in close localities. We can thus rephrase our "semantic questions" T_i in terms of the following "syntactic" queries Q_i:

- $Q1$: What sources are in the same or close localities as *amnazon.com*?
- $Q2$: What sources are syntactically associated with keyword "books"? Or attributes author and title?
- $Q3$: What are attributes syntactically associated with *amazon.com* and *a1books.com*?
- $Q4$: What are attributes in the same or close localities as with title and author?

3.3 The Mechanism: Associativity Search

We thus aim at supporting *associativity search*, in the form of queries $Q1$–$Q4$, which finds highly cooccurred sources or metadata, as a systematic facility for exploring a source repository. To realize such exploration, how should we define the functions of such associative search? We next propose several functional requirements (which will motivate our formal "operational" definition in Section 4).

Query-driven: In terms of *input*, such search should be dynamically driven by a given query. To begin with, as required by those application tasks of Section 3.1, we need to dynamically discover localities with respect to users query. As localities do not have clear boundaries, it is infeasible to build global clusters and statically assign sources to fixed clusters, as traditional clustering does. Instead, we need a query-driven mechanism to discover "local" clusters as dynamic neighbors to users query.

Further, such a query may involve multiple entities of mixed types. For instance, query $Q3$ involves multiple sources *amazon.com* and *a1books.com*. Another query may ask for "books" sources like *amazon.com*, and thus involve mixed types of entities– source and keywords– as input.

SID	Keyword	Attribute
s_1	book	author$[str]$, title$[str]$, isbn$[str]$, keyword$[str]$
s_2	textbook	author's name$[str]$, book title$[str]$, isbn$[str]$, keywords$[str]$, price: float
s_3	book	isbn$[str]$, price range: float
s_4	movie, dvd	title$[str]$, keywords$[str]$, director: str, actor$[str]$
s_5	movie, top seller	director$[str]$, actor$[str]$, title$[str]$
s_6	movie	director$[str]$, actor$[str]$, release$[str]$
s_7	flight, airline	from$[str]$, to$[str]$, leave on$[date]$, return on$[date]$, number of passengers$[int]$
s_8	ticket, reservation	from$[str]$, to$[str]$, departure date$[date]$, return date$[date]$
s_9	flight, reservation	from (airport code)$[str]$, to (airport code)$[str]$, departure date$[date]$, return date$[date]$, passengers$[int]$
s_{10}	airline, vacation	departure city$[str]$, arrival city$[str]$, depart on$[date]$, return on$[date]$, passenger$[int]$

Figure 5: A metadata repository.

Rank-based: In terms of *output*, such search should return the associated results in a ranked fashion. As localities do not have clear boundary and are of different granularity, for a particular input, the source or metadata will have different strength of associativity with others within or across localities. Therefore, we need a rank-based mechanism to return associated neighbors in the order of "closeness." We thus define the strength of associativity from an entity p (source or property) to query $\mathcal{Q}$ as $\mathcal{A}(p|Q)$. Further, as metadata involve different types of properties, and users query may ask for a particular type of properties (*e.g.*, query attributes), we thus need to rank each type of properties based on the associativity.

Putting the requirements of input and output together, we define our associative search as follows:

Problem 1 (Associative Search): Let a repository $\mathcal{R}$ contain a set of sources $\mathcal{S}$, where each source s_i is modeled by m sets of properties $s.P_1, \ldots, s.P_m$. Let $P_i = \cup_{S \in \mathcal{S}} S.P_i$ (*i.e.*, all type-i properties). We define the associative search problem as follows:

- *Input*: a query $\mathcal{Q} = (q_1, \ldots, q_k)$, where each $q_i \in \mathcal{S} \cup (\cup_i P_i)$ is a source or property entity.
- *Output*: m ranked lists $\vec{P}_i, i = 1, \ldots, m$. For each list $\vec{P}_i =< p_1, \ldots, p_l >$, $p_j \in P_i$ and $\mathcal{A}(p_{j-1}|\mathcal{Q}) \geq \mathcal{A}(p_j|\mathcal{Q})$; A ranked list of sources $\vec{S}$, similarly ordered by $\mathcal{A}(s_i|\mathcal{Q})$

To illustrate, let us introduce a running example to be used throughout the discussion of the paper:

Example 1: Consider a metadata repository $\mathcal{R}$, shown in Figure 5. The repository contains a set of 10 sources $\mathcal{S} = \{s_1, \ldots, s_{10}\}$ from 3 domains Books, Movies and Airfares. Each source is modeled with two types of properties– attributes and keywords. Overall, the repository contains 27 property entities, with 17 attributes in P_a and 10 keywords in P_k.

Consider a user Amy who wants to book airline tickets. She gives a query $\mathcal{Q}$ = {*from, to*}. Below shows an intuitive result that ranks sources and metadata based on associativity.

$\vec{S} = s_7, s_8, s_9, s_{10}, s_1, s_2, s_3, s_4, s_5, s_6$

$\vec{P}_a$ = from, to, departure date, return date, leave on, return on, departure city, arrival city, passenger, title, isbn, keyword, actor, director, author, price, release

$\vec{P}_k$ = flight, airline, reservation, ticket, vacation, movie, book, textbook, top seller, dvd ■

To address the above stated problem, there are two key challenges: first, how to define quantitatively the associativity of an entity with respect to the query; second, how to efficiently compute such associativity and rank them accordingly. In the following two sections, we will address the two issues respectively.

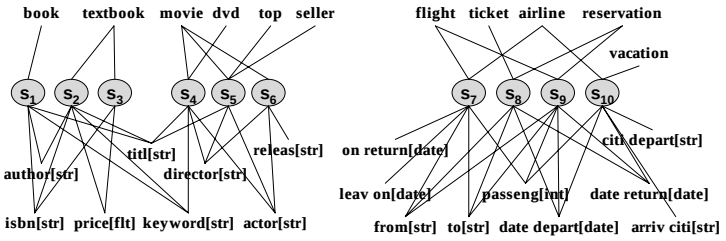


Figure 6: Data model.

4. THE FRAMEWORK: ASSOCIATIVITY AS CONNECTIVITY

To realize query-driven rank-based associativity discovery, the very first challenge is to define the measure of associativity to achieve the desired semantics. In this section, we first propose *metadata graph* as a representation of a metadata repository (Section 4.1), and then define the associativity measure to quantify the associativity based on the metadata graph (Section 4.2).

4.1 Metadata Graph

As illustrated in the previous section, associativity as direct or indirect occurrences of metadata in sources, essentially represents how things are connected through occurrences at different "orders." Two sources are first-order associated if they are directly connected by common metadata occurrence, and high-order associated if they are indirectly connected through "transitive" occurrences.

To measure the associativity of different orders, we introduce a graph-based representation of repository that embodies the associativity with physical connections in the graph. Specifically, we represent our metadata repository with a bipartite graph, which we call *metadata graph*. This bipartite graph captures source and metadata properties with two types nodes– source node and property node. There is an edge between a source entity and a property entity if and only if the property entity occurs in this source. Such a metadata graph captures associativity at different orders: the first order associativity corresponds to direct edges between nodes, and high-order associativity corresponds to paths with multiple hops. Specifically, we define the metadata graph as:

Definition 2 (Metadata Graph): Given a source repository as a set of source $\mathcal{S}$, each $s \in \mathcal{S}$ modeled with m sets of property entities $s.P_1, \ldots, s.P_m$, the metadata graph of the repository is a bipartite graph $G =< V, E >$ where $V = \mathcal{S} \cup (\cup_i P_i)$ and $E \subseteq V \times V$. There exist an edge $(s, v) \in E$ if and only if there exists a source $s \in \mathcal{S}$ such that $v \in \cup_i s.P_i$. ■

How to construct: We construct this graph representation by merging the same metadata properties from different sources. To build such a merged graph, we need to identify those "common" property entities. For instance, attribute "depart on[date]" and "departure date[date]," or keyword "books" and "book" should be viewed as common, although they are not identically the same. This is, in general, a matching problem. However, since we do not have deep understanding of sources (such as schema matching) during exploration, the techniques for merging entities thus must be simple and purely syntactical. Specifically, we employ syntax-based matching strategy using textual similarity and type similarity for attribute matching (*e.g.*, "depart on[date]" to "departure date[date]") and textual similarity for keyword matching (*e.g.*, "books" and "book"). Figure 6 shows the metadata graph of the repository in our running example, *e.g.*, "titl[str]" from s_1, s_2, s_4 and s_5 are merged.

4.2 Associativity Measure

The abstraction of repository with a graph model inspires a network-based definition for quantifying associativity. Since our associativity essentially represents how closely connected two nodes are, directly or transitively in the localities, the "connectivity" between nodes thus intuitively captures the strength of associativity. The more connected (direct or indirect) two nodes are, the stronger their associativity measure. Specifically, such associativity measure needs to have the following two properties to capture the semantic associativity.

- *Width enhancement*: That is, associativity measure should positively proportional to the *number of paths* between two nodes. The more paths connecting two nodes (*e.g.*, when two sources co-contain many common attributes), the stronger their associativity.
- *Length penalty*: That is, associativity should negatively proportional to the *length of path* between two nodes. The higher-order of co-occurrence between the two nodes are, the less associated they should be.

To capture the above two characteristics, we propose a "simulation" based approach to realize the two requirements. The simulation mimics the ripple wave effect of wave propagation (of a single source) and interference (of multiple sources) throughout the network. When there is a single source, the effect of wave propagation mimics the phenomenon of length penalty. Specifically, when wave propagates from a single origin, the magnitude of wave at a certain point is inversely proportional to the distance of this point to the origin. When spreading out the waves, a point will affect its neighbors with a decayed portion of its own magnitude, while keeping its own magnitude level. When multiple origins exist, the interference of waves mimics width enhancement. The magnitude of wave at a point is aggregated from multiple paths reaching this point. Therefore, such a ripple wave model nicely captures both of our requirements.

By overlaying this ripple simulation model over the network that captures the spreading topology, we define our associativity measure. Specifically, the simulation starts with assigning some momenta to initial query nodes (as origins of ripple waves) and let it propagate to their neighbors (through the network) progressively. Starting with query nodes with certain initial associativity (analogously wave magnitude), at first round of propagation, all their neighbors one-hop away (to the query nodes) will get associativity decayed by a factor λ. Specifically, a query node V_i with initial associativity denoted as $\mathcal{A}_0(V_i|\mathcal{Q})$ will dispatch λ portion of $\mathcal{A}_0(V_i|\mathcal{Q})$ to its neighbors, while keep its own associativity $\mathcal{A}_0(V_i|\mathcal{Q})$. Similarly in the following iterations, a neighbor node further dispatches λ portion of its own associativity to its neighbors, which are one hop farther to the query nodes (*length penalty*). When a node can be reached from multiple neighbors through multiple paths (including path of length 0), the associativity of the node is aggregated from all those paths (*width enhancement*).

Given a node with associativity $\mathcal{A}_0(V_i|\mathcal{Q})$, how will it distribute $\lambda\mathcal{A}_0(V_i|\mathcal{Q})$ to its neighbors? We observe that neighbors of different types often have different capacities to spread out associativity. Specifically, attributes bear stronger indication on semantic associativity than keywords, because keywords tend to be more ambiguous (inherently or due to the deficiency of keyword modeling). Therefore, associativity should spread more through attribute nodes than keywords. To capture the difference of capacities, we use a control parameter α to control the portion of associativity $\lambda\mathcal{A}_0(V_i|\mathcal{Q})$ distributed to a particular type of property nodes. Among neighbors of the same type, we uniformly distribute the associativity. More formally, we define our associativity measure as follows:

Definition 3: Given a metadata graph $G = (V, E)$, where $V = calS \cup P_a \cup P_k$, a query $\mathcal{Q}$, and a node $V_i \in V$ with $|V_i|$ neighbors. Without loss of generality, assume the neighbor of V_i are $V_j, j = 1, \ldots, |V_i|$. Let $\mathcal{A}_0(V_i|\mathcal{Q})$ represent the initial associativity of V_i to $\mathcal{Q}$. The associativity of V_i to $\mathcal{Q}$, denoted as $\mathcal{A}(V_i|\mathcal{Q})$ is:

$$\mathcal{A}(V_i|\mathcal{Q}) = \lambda \sum_{j=1,\ldots,|V_i|} w_{i,j} \times \mathcal{A}(V_j|\mathcal{Q}) + \mathcal{A}_0(V_i|Q) \quad (1)$$

where

$$w_{i,j} = \begin{cases} \frac{\alpha_a}{|V_j.P_a|} + \frac{\alpha_k}{|V_j.P_k|} & \text{if } V_j \in \mathcal{S} \\ \frac{1}{|V_j|} & \text{if } V_j \in P_a \cup P_k \end{cases} \quad \alpha_a + \alpha_k = 1 \quad \blacksquare$$

The above definition states that the associativity of V_i is obtained through paths of different lengths, including the associativity of from its neighbors decayed by factor λ, which corresponds to paths of length greater than 0, and the initial associativity, which corresponds to the path of length 0.

We can rewrite the above definition into a vector representation. Let $\vec{\mathcal{A}} = (\mathcal{A}(V_1|\mathcal{Q}), \ldots, \mathcal{A}(V_N|\mathcal{Q}))^T$ denote the associativity vector for nodes in V, and $W_i = (W_{i,1}, \ldots, W_{i,N})$ where $W_{i,j}$ denote the propagation coefficient of V_j to V_i, that is,

$$W_{i,j} = \begin{cases} w_{i,j} & \text{if } (V_i, V_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The definition of $\mathcal{A}(V_i|\mathcal{Q})$ in Equation 1 can be rewritten as:

$$\mathcal{A}(V_i|\mathcal{Q}) = \lambda \times W_i \times \vec{\mathcal{A}} + \mathcal{A}_0(V_i|\mathcal{Q})$$

Putting the associativity definition of all nodes together, we have the following matrix representation:

$$\vec{\mathcal{A}} = \lambda \times W \times \vec{\mathcal{A}} + \vec{\mathcal{A}}_0 \quad (2)$$

where $W = (W_1, \ldots, W_N)$ an $\vec{\mathcal{A}}_0 = (\mathcal{A}_0(V_1|\mathcal{Q}), \ldots, \mathcal{A}(V_N|\mathcal{Q}))^T$

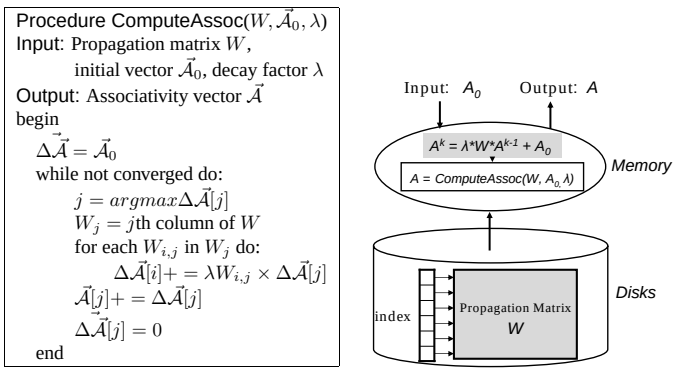
Existence of solution: Given the above matrix representation, the solution to Equation 2 is thus the associativity vector to be computed. To show that Equation 2 has a solution, let us rewrite it into the following format, by normalizing $\vec{\mathcal{A}}$ to 1:

$$\vec{\mathcal{A}} = \lambda W \vec{\mathcal{A}} + \vec{\mathcal{A}}_0 e^T \vec{\mathcal{A}} = (\lambda W + \vec{\mathcal{A}}_0 e^T)\vec{\mathcal{A}}$$

Let $W' = \lambda W + \vec{\mathcal{A}}_0 e^T$, the above equation has a solution if W' is a Markov chain. Note that, as W is a stochastic matrix, and if we can normalize $\vec{\mathcal{A}}_0$ to $1 - \lambda$, matrix W' is then a Markov chain. To normalize $\vec{\mathcal{A}}_0$ to $1 - \lambda$,, we simply scale the vector to $(1 - \lambda)\vec{\mathcal{A}}_0/|\vec{\mathcal{A}}_0|$. Note that such normalization will scale proportionally the original solution to Equation 2, and thus will not change the ranking. For simplicity, we still use $\vec{\mathcal{A}}_0$ denote the normalized vector. Given W' is a Markov chain, the solution to the above equation is thus the eigenvector of W'.

5. THE COMPUTATION MECHANISM

The eigenvector of our matrix can be iteratively computed using the Power method, or its improvements - Jacobi method and Gauss-Seidel method [9]. Briefly speaking, Power method iteratively computes associativity vector in k-th iteration denoted as $\vec{\mathcal{A}}^{(k)}$ through its $(k-1)$-th iteration till convergence. (To guarantee the convergence, the propagation matrix must be irreducible, which means that our metadata graph is a fully connected graph with paths between any pair of nodes. As we will see later, if such full associativity does not hold, we can decompose the matrix into



(a). Algorithm **(b).** Architecture

Figure 7: Basic computation framework.

several submatrices, each of which is irreducible, and compute their eigenvectors individually.), *i.e.*,

$$\vec{\mathcal{A}}^{(k)} = \lambda \times W \times \vec{\mathcal{A}}^{(k-1)} + \vec{\mathcal{A}}_0 \quad (3)$$

The power method can be reformulated in terms of the change in associativity vector. Let us define $\Delta\vec{\mathcal{A}}^k = \vec{\mathcal{A}}^k - \vec{\mathcal{A}}^{k-1}$, which can be recursively formulated as: $\Delta\vec{\mathcal{A}}^{(k)} = \lambda \times W \times \Delta\vec{\mathcal{A}}^{(k-1)}$.

We can compute $\vec{\mathcal{A}}$ by aggregating the Δ values over successive iterations. This reformulation enables large degree of flexibility in prioritization of updates[19]. In particular, in our approach, instead of updating the whole associativity vector at each iteration, we prioritize the updates of nodes based on their Δ. Updating nodes with larger Δ triggers more changes resulting in a faster progress towards convergence.

We give an outline of our algorithm in Figure 7(a). For each source, we keep track of its current associativity value $\vec{\mathcal{A}}$ and the $\Delta\vec{\mathcal{A}}$ to be propagated. In each iteration, we choose the node V_j with largest Δ to process. The processing includes two parts: updating current associativity value, and propagating the change to its neighbors. To update, we add $\Delta\vec{\mathcal{A}}_j$ to $\vec{\mathcal{A}}_j$, and reset $\Delta\vec{\mathcal{A}}_j = 0$. To propagate, for each node V_i, we increment its $\Delta\vec{\mathcal{A}}_i$ by $\lambda \times W_{i,j} \times \Delta\vec{\mathcal{A}}_j$.

Figure 7(b) shows the architecture of this computation framework. Given an initial associativity vector $\vec{\mathcal{A}}_0$ obtained from user query, our goal is to compute the final associativity vector $\vec{\mathcal{A}}$ as defined by Equation 3. This computation is done by our core algorithm ComputeAssoc($W, \vec{\mathcal{A}}_0, \lambda$) in Figure 7(a), which requires, as input, the initial vector from online user query $\vec{\mathcal{A}}_0$, a pre-configured decay factor λ and a propagation matrix W prepared offline. Given the scale of the problem, the propagation matrix W will reside on disk and be retrieved to memory at run time. At each iteration, the algorithm needs to retrieve a column in propagation matrix. To accelerate this retrieval, we construct an index structure as inverted list from nodes to the corresponding column vector.

Objective: Given the scale of problem, the propagation matrix is very large and cannot be memory-resident. This results in repeated disk accesses to the matrix and repeated matrix multiplication in memory, and therefore, the basic approach can only serve as an offline computation mechanism. To provide interactive search facility to users, we are looking for optimization strategies to speed up the convergence.

Challenge: As the architecture indicates, the bottleneck in computing $\vec{\mathcal{A}}$ is the repeated accesses to propagation matrix on the disk, and repeated matrix multiplication for propagating Δ. Therefore,

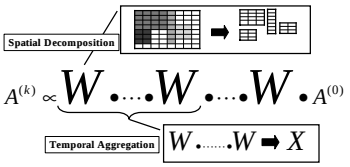


Figure 8: Spatial and temporal optimization.

the essential challenge in optimization is: How to reduce the cost of repeated matrix accesses and multiplication.

As Figure 8 shows, the definition of associativity vector in Equation 3 indicates that the iterative computation of $\vec{\mathcal{A}}$ essentially involves multiplying propagation matrix repeatedly to the initial vector to expand along different paths progressively. Therefore, to optimize, there are two dimensions, as Figure 8 intuitively shows: First, for "individual" W, the *spatial* dimension explores the spatial localities of W to decompose the big matrix into multiple smaller ones to realize a divide-and-conquer approach. By reducing the size of original matrix by a factor m, we can reduce the complexity of repeated matrix multiplication by a factor exponential to m. Second, across "multiple" W, *temporal* dimension explores the temporal repetition of matrix production to pre-compute the aggregated matrix and thus reduces the number of online multiplication of matrices.

5.1 Spatial Optimization: Matrix Locality

To optimize along the spatial dimension, we want to "localize" the accesses to matrix W to a small region, which can largely fit into memory without repeatedly being retrieved from disks. In this section, we discuss three stages towards such localization.

Matrix Decomposition: The bipartite nature of our metadata graph gives us the first opportunity to localize the access. Note that, since our metadata graph is bipartite, the propagation matrix has non-zero entries only for source-attribute combination and source-keyword combination. Let us order vector $\vec{\mathcal{A}}$ to make nodes of the same type adjacent, *i.e.*, $\vec{\mathcal{A}} = (\vec{S}^T, \vec{K}^T, \vec{A}^T)^T$, $\vec{\mathcal{A}}_0 = (\vec{S}_0^T, \vec{K}_0^T, \vec{A}_0^T)^T$, where $\vec{S}$, $\vec{K}$ and $\vec{A}$ are source, keyword and attribute associativity vector respectively, and $\vec{S}_0, \vec{K}_0$ and $\vec{A}_0$ their initial associativity vector. By applying this re-ordering, we partition the propagation matrix into four non-zero submatrices W_{SK}, W_{SA}, W_{KS} and W_{AS}. Equation 2 can now be rewritten as:

$$\begin{pmatrix} \vec{S} \\ \vec{K} \\ \vec{A} \end{pmatrix} = \begin{pmatrix} 0 & W_{SK} & W_{SA} \\ W_{KS} & 0 & 0 \\ W_{AS} & 0 & 0 \end{pmatrix} \begin{pmatrix} \vec{S} \\ \vec{K} \\ \vec{A} \end{pmatrix} + \begin{pmatrix} \vec{S}_0 \\ \vec{K}_0 \\ \vec{A}_0 \end{pmatrix}$$

We can further rewrite the above equation into three equations, which define how to compute $\vec{S}$, $\vec{A}$ and $\vec{K}$ respectively as follows:

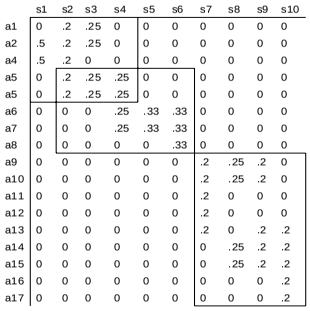
$$\vec{S} = \lambda W_{SK} \times \vec{K} + \lambda W_{SA} \times \vec{A} + \vec{S}_0 \quad (4)$$

$$\vec{A} = \lambda W_{AS} \times \vec{A} + \vec{A}_0 \quad (5)$$

$$\vec{K} = \lambda W_{KS} \times \vec{S} + \vec{K}_0 \quad (6)$$

By such decomposition, we avoid unnecessary matrix multiplication between the zero-submatrix and $\vec{\mathcal{A}}$. Specifically, we reduce 9 matrix multiplications (corresponding to all combinations of source, attribute and keywords) to 4 (corresponding to source-keyword and source-attribute combinations).

Top-K Pruning: We further explore the localities within submatrices W_{AS}, W_{SA}, W_{KS} and W_{SK}. The motivation of such optimization is the observation that accesses to the propagation vector of each node can be localized to a small portion of entries. As the



	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
a1	0	.2	.25	0	0	0	0	0	0	0
a2	.5	.2	.25	0	0	0	0	0	0	0
a4	.5	.2	0	0	0	0	0	0	0	0
a5	0	.2	.25	.25	0	0	0	0	0	0
a5	0	.2	.25	.25	0	0	0	0	0	0
a6	0	0	0	.25	.33	.33	0	0	0	0
a7	0	0	0	.25	.33	.33	0	0	0	0
a8	0	0	0	0	0	.33	0	0	0	0
a9	0	0	0	0	0	0	.2	.25	.2	0
a10	0	0	0	0	0	0	.2	.25	.2	0
a11	0	0	0	0	0	0	.2	0	0	0
a12	0	0	0	0	0	0	.2	0	0	0
a13	0	0	0	0	0	0	.2	0	.2	.2
a14	0	0	0	0	0	0	0	.25	.2	.2
a15	0	0	0	0	0	0	0	.25	.2	.2
a16	0	0	0	0	0	0	0	0	0	.2
a17	0	0	0	0	0	0	0	0	0	.2

Figure 9: Blocks in propagation matrix W_{AS}.

occurrence localities in our survey of Section 3 indicates, a source only contains a small number of attributes, and an attribute only occurs in a small number of sources. This indicates that, given j-th column of propagation matrix W_{SA}, representing the propagation coefficients of attribute a_j, after several iterations, there are only a very small portion of the vector contains significant values, while others are either zero or close to zero. Similar observation holds for other matrices. This means that the access to a column of propagation coefficients can be localized to a small portion of significant entries.

Taking advantage of such access locality within a column, we propose to localize the access to only the top-k most significant neighbors of a node. This effectively prunes each column in the matrix to only the top-k entries. As a realization of the basic architecture, top-k pruning approach retrieves only the pruned column at each iteration to compute Equation 4, 5 and 6.

Block-based Update: While the top-k pruning strategy reduces the size of matrix to be retrieved from disk, our experimental results show the noticeable accuracy sacrifice. The reason is that the choice of k in the top-k approach is rather heuristic. Fixing a universal k value for all nodes is not appropriate because different nodes may have different numbers of significant neighbors. A small k value will miss many relevant neighbors, while a big k value incurs big space overhead. Further, while the top-k pruning strategy takes advantage of the access locality within individual columns of propagation matrix, it does not consider the localities revealed across different columns.

As our survey of Figure 4 shows, there are localities between sources and attributes: a small set of sources in a domain often contains a small set of vocabularies of that domain. Such localities of occurrences are reflected as "blocks" of non-zero (or significant values to be more precise) entries located along the diagonal line when matrix is properly ordered. Figure 9 shows such blocking phenomenon in matrix W_{AS}. Specifically, those blocks are densely populated with non-zero values and different blocks are minimally overlapped. Such block behavior suggests that the locality exists not only within individual columns but also across different columns, which thus gives us further opportunity to optimize. As different columns within the same blocks often share the same attributes, those columns are likely to be accessed successively, and therefore should be arranged together on the disk.

To leverage such localities in the propagation matrix, we want to reorder the propagation matrix to formulate the blocks. An ideal reordering is to obtain submatrices along the diagonal line, which do not overlap at all with each other. However, in practice, such a perfect decomposition of the propagation matrix rarely exists, because of the fuzzy localities as we observed in our survey.

Although perfect decomposition may not exist, a "good" decomposition can still help to localize the accesses greatly. Such a good decomposition should decompose the original matrix into multiple submatrices, which satisfy the following two properties: First, *denseness*: each submatrix should be densely populated with significant values; Second, *disjunctness*: different submatrices overlap as less as possible. Figure 9 shows such a decomposition for matrix W_{AS}.

Note that, as submatrices have some overlaps, such a decomposition results in redundancy in repeatedly storing matrix entries shared by different submatrices. However, such overlap have the advantage that it can localize propagation only within the blocks where query nodes are located, without propagating across blocks. The reason is that, by allowing overlaps, a submatrix bounds all significant neighbors that can be reached during propagation, and therefore computation does not need to go across different blocks.

Such block-based update helps in improving the efficiency of our computation. First, since entries in those diagonal matrices are frequently accessed, we can load those diagonal matrices into memory if the space allows. Second, even fully loading is not possible, given the access locality, we know successive accesses to the matrix tend to localize to a particular block, and thus random disk accesses can be minimized.

To realize the decomposition of matrix into blocks, there are various existing techniques. The problem of matrix reordering has been studied with various objectives. [11] studies how to reorder a sparse matrix for fast graph partitioning. [18] studies how to reorder boolean matrix for compressing matrix for efficient storage. [10] studies how to partition the sources using term-document occurrence matrix. In our implementation, we develop a *hyperclique* [16] based blocking which partitions the sources into groups of sources that share a common set of vocabulary. The partitioning of sources thus obtained is then used for decomposition of propagation matrix. Due to space limitation, we omit the details of hyperclique based blocking.

Why not clustering: As a remark, while we use clustering techniques for decomposing the matrix into blocks, we would like to note that they cannot be used as complete solution to our problem. To begin with, clustering only partially address our task T_1 not the others. Second, clustering can provide only static organization of sources, while we want to provide a query driven dynamic exploration facility. Third, we use the blocking only as an optimization strategy for improving efficiency. While "good" partitioning will improve the computation efficiency, "bad" partitioning will not hurt the semantic accuracy.

5.2 Temporal Optimization: Matrix Pre-aggregation

This section further discusses optimization strategies along the temporal dimension to reduce the number of times for matrix retrieval and multiplication.

The computation approaches discussed so far basically follows "one-hop at a time" propagation paradigm. As Equation 4, 6 and 5 indicates, at each iteration, the basic algorithm propagates associativity from source on-hop to its neighbor keywords and attributes, which in next iteration propagates on-hop back to the neighbor sources. We thus ask: whether we can expand "multiple-hops" at a time so that we retrieve and multiply the matrix less number of times to reach convergence. The two approaches we discuss in this section will explore this idea to realize optimization along temporal dimension. As Figure 8 suggests, the basic approach is to pre-aggregate matrix offline to compute multiple-hops propagation coefficients and initial associativity. By such offline precomputation, we reduce online matrix multiplication time to achieve better response time.

Two-hops Aggregation: The first optimization is to aggregate the two-step propagation from source to attribute (keyword) and attribute (keyword) to source into one step propagation of source to source. Let us consider the definition of vector $\vec{S}$, $\vec{A}$ and $\vec{K}$ in Equation 4, 5 and 6. As those vectors are defined in a mutual recursive way, we can rewrite the definition of $\vec{S}$ in Equation 4 by substituting $\vec{K}$ and $\vec{A}$ using Equation 5 and 6.

$$\begin{aligned}\vec{S} &= (\lambda^2 W_{SK} \times W_{KS} \times + \lambda^2 W_{SA} \times W_{AS}) \times \vec{S} \\ &+ (\lambda W_{SK} \times \vec{K^{(0)}} + \lambda W_{SA} \times \vec{A^{(0)}} + \vec{S^{(0)}})\end{aligned}$$

For simplicity, let $M = \lambda^2 W_{SK} \times W_{KS} + \lambda^2 W_{SA} \times W_{AS}$, and $Q_0 = \lambda W_{SK} \times \vec{K^{(0)}} + \lambda W_{SA} \times \vec{A^{(0)}} + \vec{S^{(0)}}$. We thus have:

$$\vec{S} = M\vec{S} + Q_0 \quad (7)$$

Applying the same algorithm in Figure 8(a) with propagation matrix M and initial vector Q_0, we proceed two steps at a time, propagating associativity from source to source directly without going through intermediate attributes or keywords. This transforms the repeated accesses to the four matrices to accesses to M only. The four matrices are accessed only twice: once at the pre-computation time to compute M and once at end of online computation to compute $\vec{A}$ and $\vec{K}$ using Equation 5 and 6.

Multiple-hops aggregation: Applying the same idea, we can perform multiple-hops propagation to expand from a source to its length-l neighbors. As Equation 7 suggests, the associativity vector $\vec{S^{2l}}$ at $2l$-th iteration can be defined upon that at l-th iteration as follows::

$$\vec{S^{(2l)}} = M^l \times \vec{S^{(l)}} + (\sum_{i=0,\ldots,l-1} M^i) \times Q_0 \quad (8)$$

Let $X = M^l$ and $L_0 = (\sum_{i=0,\ldots,l-1} M^i)$. This equation states that the associativity vector at step $2l$ can be obtained through the vector at step l using propagation matrix X and initial vector $L_0 \times Q_0$. To generalize, we can verify that the following equation holds:

$$\vec{S^{(kl)}} = X \times \vec{S^{(k-1)l}} + L_0 \times Q_0 \quad (9)$$

The equation states that the associativity vector at step kl can be obtained through the vector at step $(k-1)l$ using X and $L_0 \times Q_0$. Therefore, we effectively proceed l steps at a time towards computing $\vec{S}$, which speed up the computation. As a realization of the architecture, the precomputation approach applies the same algorithm using propagation matrix X and initial vector $L_0 \times Q_0$.

Discussion on speed up: Although intuitively such precomputation should linearly speed up the computation by "striding" multiple steps at a time, in practice, we cannot obtain such a speed up. The reason is two folded: First, the temporal optimized approach has non-negligible startup time. Unlike the basic progressive approach, the multiple-hops propagation needs to convert the initial associativity vector $\vec{A_0}$ to $L_0 \times Q_0$, which involves matrix multiplications. Second, the precomputation may potentially lose the benefit of early termination of propagation. Since precomputation of m-hops essentially aggregates all paths within length m, it may overdo for some sources, which can stop before m iterations in basic approach.

6. EXPERIMENTAL EVALUATION

We now study the effectiveness of our framework and the speed up obtained by several optimization techniques.

6.1 Evaluation Environment

Dataset: We use two datasets for our experiments. First is a manually collected dataset of about 500 sources in 8 domains - Airlines, Automobiles, Books, Car-Rentals, Hotels, Jobs, Movies and Music Records. This dataset, which we shall refer to as *small-dataset*, is available at *http://metaquerier.cs.uiuc.edu*, as part of the *UIUC Web Integration Repository*. Second dataset has about 30K sources obtained using a focused deep web crawler [14]. We combine sources in *small-dataset* to these crawled sources to obtain the *large-dataset*.

To obtain attribute metadata information, we use the form extraction techniques [26] on these sources. We apply stemming, stop word removal, and alphabetic re-ordering on the attribute names to obtain labels for normalized representation. To obtain keyword information, we apply stemming and stopword removal on the texts extracted from the home pages and query interface pages of the sources. For every source, we choose the 10 most frequent keywords, with a restriction that any selected keyword must occur in at least 0.3% of sources.

Benchmark Queries: We create a set of 240 queries similar to the queries we discussed earlier in Section3.2. These *benchmark-queries* seek sources in one of the 8 domains. For each of these queries, we use the sources from *small-dataset* in the corresponding domain as the *ground truth*, and evaluate the effectiveness of our framework using precision-recall metric.

Implementation Details: We perform all our experiments on 2.60 GHz hyper-threaded P4 machine with 1 GB memory. The metadata extraction for 30K sources takes about 2-3 days. Preparing the propagation matrices from extracted metadata takes less than an hour. For fast access to propagation matrices, we use BTrees in python bsddb module.

6.2 Effectiveness of the Framework

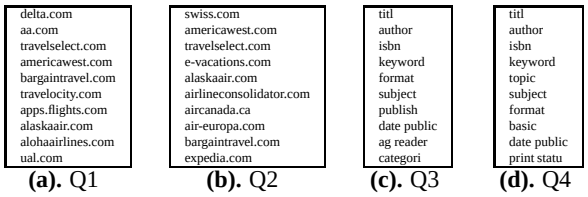
In this section, we demonstrate the usefulness of our framework in answering various semantic questions. We show the results obtained for the queries we asked earlier in Section 3, and also quantitatively evaluate the effectiveness under precision/recall metric for the *benchmark-queries*.

Illustrative results: As the first scenario, consider end users who want to find airline sources. Such users can use our framework by asking queries, such as: $Q1 = \{$*aa.com*$\}$ or $Q2 = \{$*flight, airfare*$\}$. The top 10 results for the two queries are illustrated in Figure 10 (a) and (b), respectively. We find that both these results are all airline sources. In fact, 25 out of the top 30 sources obtained for the two queries are same.

As a second scenario, consider an integration system developer who wants to find the typical attributes used by sources in book domain. Such a user gives a query $Q3 = \{$*amazon.com, a1books.com*$\}$. For the same task, another developer gives a query $Q4 = \{$*fiction, bookstore*$\}$. The top 10 attributes for the two queries are shown in Figure 10 (c) and (d), respectively. We find that the results for the two queries are not only impressive, but also very similar. In fact, some of the attributes in results of $Q3$ do not appear in any of the sources in that query.

Precision on benchmark queries: We compare the sources obtained using the *baseline* approach for *benchmark-queries* against the ground truth. We find that it achieves a mean average precision of 82%, quantitatively showing the high effectiveness of our approach.

Necessity of multiple schematic metadata: We use multiple types of schematic metadata for two reasons: (a) The user can ask queries



(a). Q1 **(b).** Q2 **(c).** Q3 **(d).** Q4

Figure 10: Top results for illustrative queries.

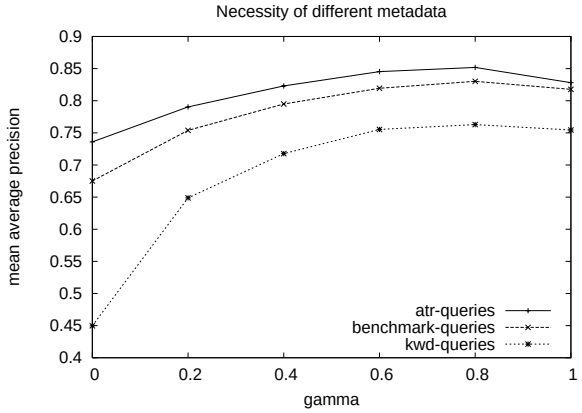


Figure 11: MAP for *baseline* with variation in gamma

using different types of metadata or seek as results different metadata, and (b) When not directly used, our model can still leverage the presence of different types of metadata for improving the accuracy of results. Here, we quantitatively evaluate the second reason.

A source node propagates a fraction, $\gamma \in [0, 1]$, of its associativity to attributes, while the remaining $(1 - \gamma)$ to keywords. By varying γ, we can study the necessity of both attributes and keywords in our model. As shown in Figure 11, the mean average precision (map) of our model on the *benchmark-queries* shows an inverted-U shaped curve, indicating that both attributes and keywords contribute to optimal performance.

As benchmark-queries contains both attributes and keywords, we want to further study the necessity of keywords when the queries contain only attributes, and vice-versa. We create a set of 110 queries, *atr-queries*, containing only attributes. As we can see in Figure 11, the performance still exhibits a inverted-U shaped curve. We observe similar phenomenon for *kwd-queries*, a set of queries containing only keywords.

We observe that the best performance is obtained when γ is 0.8, indicating that the associativity of attributes is semantically more meaningful than the associativity of keywords. We believe that with more sophisticated keyword selection [25], we will find a greater role for keywords.

Effect of degree of propagation: We study the effect of parameter λ, which controls the degree of associativity propagation, on the effectiveness of our approach. With variation in values of λ from $[0.05, 0.95]$, we observe only a marginal variation in mean average precision on *benchmark-queries*. The propagation even with high decay factor seems to be quite effective.

The reason for such a counter-intuitive observation is that our *benchmark-queries* seek sources in a very specialized domain. Even a small amount of propagation is sufficient to explore such a narrow domain. The applications that require sources in a broader domain, e.g., travel related sources, entertainment sources, will find a greater effect of the degree of propagation. For example, for a query that seeks travel related sources, we should return sources in Airfare, Car Rentals and Hotels domain but not from Automobile domain.

method	accuracy-convergence	Time-for-convergence
baseline	100 %	1.0 s
top5	31 %	0.1 s
top8	55 %	0.2 s
top10	65 %	0.4 s
hc-block	98%	0.6 s

Figure 12: Comparision of spatial optimization techniques

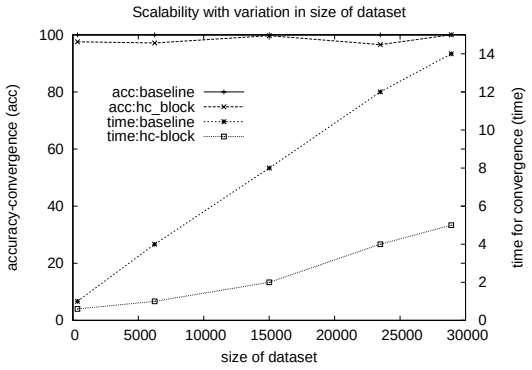


Figure 13: Scalability with variation in size of dataset

6.3 Computational Efficiency

Having studied the semantic effectiveness of our framework, we now evaluate its computational efficiency. We evaluate our spatial and temporal optimization techniques against *baseline* approach along two dimensions: (a) what is the speed up, and (b) how approximate are the results. To measure the approximation in results, we use a new metric - *accuracy-convergence*, which we define as the ratio of observed *MAP* to the *MAP* obtained by *baseline* algorithm upon convergence.

Evaluation of spatial optimization techniques: We compare the top-k strategy with $k = 2, 5, 8$ and the blocking strategy - *hc-block* against the *baseline* method using the benchmark queries on *small-dataset*. Figure 12 shows the *accuracy-convergence* and the time for convergence for different methods. We find that top-k methods provide significant speed up over *baseline* method, however, very low values of *accuracy-convergence* suggests that in providing such a speed up these methods tend to degrade the quality of results. In contrast, the *hc-block* method is also able to provide significant speed up and still achieves the *accuracy-convergence* of nearly 100%. Next, we will study the robustness of the performance benefit of *hc-block* as we increase the size of dataset.

Scalability with increase in size of dataset To study the performance of *hc-block* against *baseline* approach with variation in size of dataset, we use sources obtained from large scale crawling. We sort the sources gathered from large scale crawling in the alphabetical ordering of the URL and incrementally add them to the *small-dataset*. Using the benchmark queries, we compare the *accuracy-convergence* and time for convergence of *hc-block* against the *baseline* method.

As shown in Figure 13, for all sizes of dataset *hc-block* is able to achieve the *accuracy-convergence* of nearly 100%. More interestingly, the time for convergence for *hc-block* is less than half of the *baseline*, showing a speed up by at least a factor of 2. Further, the increase of time for convergence with increase of size of dataset is slower for *hc-block* as compared to baseline. This suggests that not

	Th_L		
n	1.001	1.0001	1.00001
2	0.22	0.51	4.30
5	0.20	3.21	3.05
8	0.19	2.87	2.78

Figure 14: Speed up of temporal optimization

only *hc-block* significantly speeds up the computation, but it is also more scalable than the *baseline* approach.

Speed up using temporal optimization We saw that for all dataset sizes *hc-block* speeds up the computation by a factor of 2 and still achieves nearly 100% *accuracy-convergence*. We now study if we can further improve the performance of *hc-block* using temporal optimization. There are two possible implementations of temporal optimization. In first approach, we compute $L_0 \times Q_0$ and input query, and use it as the final result. Alternatively, we can apply further propagation using the precomputed matrix *X*. We believe that if n, which controls the degree of propagation, is large enough further propagation would be unnecessary. Therefore, in our implementation we compute $L_0 \times Q_0$, and present the results thus obtained to user.

We find that the matrix L_0 is very dense. However, if we prune the entries in each of the column of L_0 that are below a certain threshold, say Th_L, we can obtain significant speed up. Larger the value of Th_L, more aggressive is the pruning. However, we observe that as long as Th_L is small enough, its choice does not affect the *accuracy-convergence*.

In Figure 14, for different values of n and Th_L, we show the ratio of response time of *hc-block* without temporal optimization to the response time with temporal optimization on *large-dataset*.

For all cases in Figure 14, the *accuracy-convergence* for *hc-block* with temporal optimization is about 83%, which suggests that the value of n does not affect the quality of results. While this may seem counter-intuitive, we explained the reason for this phenomenon earlier in Section 6.2. We believe that queries exploring broader domains, e.g., travel related sources, entertainment sources, would show greater dependency on the degree of propagation.

For all values of n, smaller the value of Th_L greater is the computational speed up. Infact, with $Th_L = 1.001$ and $n = 2$, temporal optimization reduces the response time to about 23%, a significant speed up of about 4.4 times.

Recall that *hc-block* already performs more than twice as fast as the *baseline* approach. Thus, overall, using our spatial and temporal optimization strategies, we can achieve a speed up by a factor of about 10 times over the *baseline* approach.

7. CONCLUSION

This paper aims at the new problem of metadata-based exploration for guiding access and integration of the deep Web. With massive and diverse sources proliferating online, we are lacking an effective facility for finding "where sources are" and "what vocabularies they speak." Towards this goal, we propose: (a) Schematic metadata model for describing sources; (b) Query-driven rank-based associativity search mechanism. Our techniques hinge on a novel transformation from cooccurrence analysis to link analysis, thus enabling a systematic and unified framework for searching associativity between any (source and metadata) entities.

Our experiments on real world large-scale dataset demonstrate high effectiveness. Realizing this framework for an interactive exploration is very challenging. We exploit the characteristics of deep Web sources to develop several spatial and temporal optimization techniques. Our experiments showed that these techniques can achieve significant speedup without sacrificing the effectiveness.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[2] D. M. Alon Y. Halevy, Michael J. Franklin. Dataspaces: A new abstraction for information management. In *DASFAA*, 2006.

[3] M. K. Bergman. The deep web: Surfacing hidden value. Technical report, BrightPlanet LLC, Dec. 2000.

[4] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the web: Observations and implications. *SIGMOD Record*, 33(3):61–70, 2004.

[5] K. C.-C. Chang, B. He, and Z. Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR*, 2005.

[6] F. Crestani. Application of spreading activation techniques in information retrieval. In *Artificial Intelligence Review*, 1997.

[7] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[8] C. A. Dyer. The stroboscopic ripple tank as a teachi aid. *AJP*, (5):208–210, 1937.

[9] G. Golub and C. F. V. Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[10] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[11] A. Gupta. Fast and effective algorithms for graph partitioning and sparse matrix ordering. *IBM Journal of Research and Development*, 41(1/2):171–184, 1997.

[12] A. Y. Halevy, J. Madhavan, and P. A. Bernstein. Discovering structure in a corpus of schemas. *IEEE Data Eng. Bull.*, 2003.

[13] B. He, K. C.-C. Chang, and J. Han. Discovering complex matching over web query interfaces: a correlation mining approach. In *KDD*, 2004.

[14] B. He, C. Li, D. Killian, M. Patel, Y. Tseng, and K. C.-C. Chang. A structure-driven yield-aware web form crawler: Building a database of online databases. In *(under review)*, 2006.

[15] H. He, W. Meng, C. T. Yu, and Z. Wu. Wise-integrator: An automatic integrator of web search interfaces for e-commerce. In *VLDB*, pages 357–368, 2003.

[16] Y. Huang, H. Xiong, W. Wu, and Z. Zhang. A hybrid approach for mining maximal hyperclique patterns. In *Proceedings of the 16th IEEE Int'l Conf. on Tools with Artificial Intelligence*, 2004.

[17] G. Jeh and J. Widom. Scaling personalized web search. In *In Proc. of 12th WWW*, 2003.

[18] D. S. Johnson, S. Krishnan, J. C. abd Subodh Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. 2004.

[19] F. McSherry. A uniform approach to accelerated pagerank computation. In *WWW*, 2005.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[21] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *VLDB*, 2001.

[22] J. Wang, J. Wen, F. Lochovsky, and W. Y. Ma. Instance-based schema matching for web databases by domain-specific query probing. In *VLDB*, 2004.

[23] W. Wu, C. T. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *ACM SIGMOD*, pages 95–106, 2004.

[24] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *ACM SIGIR*, 1996.

[25] Y. Yang and J. O. Pederson. A comparative study on feature selection in text categorization. *ICML*, 1997.

[26] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *ACM SIGMOD*, 2004.