Data-based Research at IIT Bombay

Soumen Chakrabarti Ganesh Ramakrishnan
Krithi Ramamritham Sunita Sarawagi S. Sudarshan
Indian Institute of Technology Bombay
Mumbai 400076 India
{soumen,sunita,ganesh,krithi,sudarsha}@cse.iitb.ac.in

1. OVERVIEW

The Indian Institute of Technology (IIT) Bombay has a history of research and development in the area of databases, dating back to the early 1980s. D. B. Phatak and N. L. Sarda were among the first faculty members at IIT Bombay to work in the area of database systems. This was a period when the financial sector of India, headquartered primarily in Bombay (now renamed Mumbai) saw a spurt in computerization, and IIT Bombay faculty played a leading role as consultants for database implementations in these companies. Research in the area of databases began in the early 1980s, but increased greatly from the early 1990s, with the hiring of several faculty including S. Seshadri, S. Sudarshan, and later Krithi Ramamritham, who moved to IIT Bombay from U. Mass. Amherst in the early to mid 1990s. With the hiring of Sunita Sarawagi and Soumen Chakrabarti in the late 1990s, there was a significant broadening, with the group no longer being just a database group, but rather a much broader data management group, with interests in information retrieval, and data mining. More recently Ganesh Ramakrishnan joined our group, further increasing its strengths in information retrieval and data mining.

The number of PhD students increased from around 1 or 2 enrolled at a time in the early 1990s, to about 12 to 15 students at a time in recent years. While this number is much better than earlier, and is increasing rapidly, it is still small by most standards. However, our master's and bachelor's students have compensated for the shortage of PhD students, and have made very significant contributions to our research efforts, with well over three fourths of our papers having such students as coauthors.

Today, the group covers a diverse range of interests, which you can see from the different research projects showcased in this article. In the following sections, we outline the major research projects of the group. We wrap up the article with a summary of other contributions to the community, by group members.

For more information about the group, please visit:
http://www.cse.iitb.ac.in/infolab

2. RANKING IN GRAPH DATA MODELS

Graph data models are ubiquitous in semistructured search. Modeling a data graph as an electrical network, or equivalently, as a Markovian "random surfer" process, is widely used in applications that need to characterize some notion of graph proximity. Edges in most data graphs have rich semantics. The probability of a random surfer exiting a node through an outbound edge should be influenced by the semantics of the edge. In applications, relative edge conductances used to be tuned by trial and error, from domain knowledge. We gave several formulations [2, 1] to automatically learn relative edge conductances from pairwise preferences between nodes.

Random walks in weighted graphs is a form of personalized PageRank. In applications, the edge conductances are known offline, but the teleport vector is a function of the query. For large graphs, a global PageRank computation in response to every new teleport is prohibitive. We [13] proposed a query workload-driven framework to carefully choose a small number of basis nodes where certain indices are built, so that provably correct top-k PageRank nodes can be reported within a time that, in practice, remained essentially *constant* even as the graph scaled substantially.

3. LARGE-SCALE WEB ENTITY ANNOTATION AND INDEXING

A recent study by a search company revealed that up to 40% of Web queries pertain to a named entity. Search engines have been steadily supplementing the "ten blue links" with structured knowledge about entities and relationships. Toward this end, we undertook to annotate token spans in a large scale Web corpus (500 million pages) with disambiguated mentions of any Wikipedia entity (at that time, numbering between 2 and 3 million). Note that we are not interested in determining the broad type of an entity, such as person, place, time or event. We are interested in pinpointing the exact entity that has been mentioned, in the face of aliases ("John Smith") and other ambiguity ("apple").

We proposed [31] and demonstrated that collective disambiguation of mentions on a page can improve accuracy. E.g., "Michael Jordan" on a page by itself is more often about the basketball player than the Berkeley professor, and "Stuart Russell" is often the actor, not the Berkeley professor. But a document that mentions *both* "Michael Jordan" and "Stuart Russell" is almost certainly describing the Berkeley professors.

Another challenge was the design of practical data structures for fast disambiguation. Part of the statistical disambiguation model above consists of a map from a key consisting of a phrase ID p (where a phrase may be "Michael Jordan" or "big Apple"), a candidate entity ID e (the basketball player, the Berkeley professor, New York City, Apple Computer Corporation — a standard and unique URN in Wikipedia), and a context feature f (e.g., the words *variational*, *league*, or *iOS*), to a value w that is a trained model weight. The number of keys is large but the key space (p, e, f) is exceedingly sparse: each of p, e, f runs into tens of millions of distinct values. In our modest 500-million page, 2-million entity prototype, there were 700 million (p, e, f) keys, which would have cost us over 20GB of RAM using conventional Java hash maps. Lossy maps as used in signed or hash kernels showed considerably poorer accuracy than our disambiguator with lossless model weights. We designed a new workload-driven lossless compression scheme [12] that traded space for access time, cutting the $(p, e, f) \rightarrow w$ map down to only 1.2GB of RAM. At the same time, our annotator was orders of magnitude faster than other public annotators.

4. CONSENSUS-BASED QUANTITY AND ENTITY SEARCH

While commerce verticals serve plenty of quantitative information in response to Web queries, they do not cover many quantity information needs. E.g., What is the typical battery life on an iPad while playing games? What is the driving time between Mumbai and Pune? How far should I relocate a raccoon? What is the typical production budget of French art movies? Unlike record extraction to precise schema, snippets that may carry the desired information are matched very noisily, and extracting the quantity (with unit) is also nontrivial. If we plot the snippet match score against the candidate quantity embedded in the snippet, the poor reliability of snippet scores shows up as substantial vertical spread of both relevant and irrelevant snippets. However, a new signal comes to our rescue [5]: if there is palpable consensus on the answer, the relevant snippets appear in narrow vertical bands in the 2d plot. We designed new, practical quantity-ranking functions for this class of queries, and their accuracy was substantially better than prior art.

Similar issues of detecting consensus arise in regular entity search as well. Based on our 500 million-page Web corpus with over 8 billion indexed entity annotations, we have built an entity search engine [14] based on discriminative learning-to-rank models. Our system has a semi-structured query model where types from Wikipedia or YAGO can be specified, e.g., find entities belonging to `WikiCategory: Austrian_Physicist` who played the violin. However, users will generally not be familiar with the hundreds of thousands of types registered in Wikipedia, YAGO, or Freebase. They will ask the usual "telegraphic" queries (e.g., `austria physicist violin`) and expect that the search engine will divine their intention. We proposed [44] generative and discriminative approaches to simultaneously explore multiple interpretations of the query and score response entities, effectively aggregating over these interpretations to rank entities. The resulting ranking accuracy is significantly better than trying to interpret the target type first and then execute a structured query.

5. KEYWORD QUERIES ON STRUCTURED DATA

In recent years there has been a good deal of research in the area of keyword search on structured and semi-structured data, including our own work on the BANKS system [6, 29]. Most of this body of work has a significant limitation in the context of enterprise data, since it ignores the application code that has often been carefully designed to present data in a meaningful fashion to users. In recent work [40], we considered how to perform keyword search on enterprise applications. Such applications provide a number of forms that can take parameters; parameters may be explicit, or implicit such as the identifier of the user. In the context of such applications, the goal of keyword search is as follows: given a set of keywords, retrieve forms along with corresponding parameter values, such that result of each retrieved form executed on the corresponding retrieved parameter values will contain the specified keywords. For example, suppose a university ERP system includes a form that takes a roll number as parameter, and returns information about the student with that roll number. Given a keyword query "Krithi CS" our system would return (amongst other results) the above form, along with roll numbers for which the form result includes the keywords "Krithi" and "CS".

Some earlier work in this area was based on creating keyword indices on form results, but there are problems in maintaining such indices in the face of updates. In contrast, in our work we introduced techniques based on creating "inverted SQL queries" from the SQL queries in the forms. Unlike earlier work, our techniques do not require any special purpose indices, and instead make use of standard text indices supported by most database systems. We have implemented our techniques and show that keyword search can run at reasonable speeds even on large databases with a significant number of forms.

We have also been extending the BANKS system to support querying on annotated textual data; in this context, an annotation marks a particular word or phrase (e.g. Gandhi) with the entity that it (likely) refers to (e.g. the Indian leader M. K. Gandhi). Queries on such annotated textual data can search for entities (entity queries) or entities which satisfy specified relationships (entity-relationship queries). We have built a prototype that works on Wikipedia data (described in [4]), and are currently extending it to work on annotated Web crawl data.

6. WORLD WIDE TABLES (WWT)

The Web today comprises of billion of semi-structured objects such as tables and lists that have been universally accepted as an idiom for expressing relational data even for human consumption. Usually, these are considerably higher quality than completely unstructured free-format text. The goal of the WWT project is to explore methods to exploit tables and lists on the Web for various query-driven structure extraction tasks. We have explored the following challenges in the context of this project.

Extracting structure from lists.

The user poses a query by providing a few seed multi-attribute rows in a table, and expects as answers more rows of the same type. We assemble such tables on-the-fly from the few seed rows by aligning, segmenting, and consolidating information from raw lists on the Web. We deploy statistical methods such as Semi-Markov Conditional Random Fields (CRFs) [42] for this task. Our setup differs from conventional deployment of Semi-CRFs in two ways: First, we do not possess any explicitly labeled data for training purposes. Our only supervision is in terms of a few seed structured records. In [24] we report how to carefully label unstructured list items using the seed record set. Second, we need to train multiple extractors, one for each list source. We exploit the fact that the lists often enjoy partial content overlap to jointly train the model so as to ensure agreement in the labels of overlapping content [25].

Answering Column Keywords Queries.

The user poses a query consisting of keywords describing each column in a table he expects to see in the answer: example "university" and "motto". WWT returns a multi-column table in response to such queries by exploiting the millions of existing tables on the Web. We represented this task as a graphical model that jointly maps all tables by incorporating diverse sources of clues spanning matches in different parts of the table, corpus-wide co-occurrence statistics, and content overlap across table columns. We defined a novel query segmentation model for matching keywords to table columns, and a robust mechanism of exploiting content overlap across table columns. We designed an efficient inference algorithms based on bipartite matching and constrained graph cuts to solve the joint labeling task. More details of this work can be found in [37].

Annotating Web tables to an Ontology.

Most web tables are not organized on any formal, uniform schema; consequently Web search cannot take advantage of these high-quality sources of relational information. In the presence of a popular Ontology, such as Wikipedia and its derivatives like Yago, we enrich raw Web tables by linking them with the Ontology. We developed new machine learning techniques to annotate table cells with entities that they likely mention, table columns with types from which entities are drawn for cells in the column, and relations that pairs of table columns seek to express. We proposed [32] a new graphical model for making all these labeling decisions for each table simultaneously, rather than make separate local decisions for entities, types and relations. These annotations are invaluable for retrieving tables based on column keyword queries and integrating information across isolated tables.

Current status of the project.

The WWT system currently runs over a crawl of 36 million tables and lists extracted from an offline crawl of 0.5 billion Web pages. The project was started in 2008 and after five years it is nearing completion. It provided us an exciting platform to carry out research on large-scale information extraction and integration while grappling with the noise and redundancy that is so prevelant over Web data. More information is available at the project page at `http://www.cse.iitb.ac.in/ sunita/wwt`.

7. AGGREGATING IMPRECISE DATA

The goal of this project is to develop statistical learning models for extracting precise aggregate statistics over sets of instances. We highlight three diverse facets for this task:

Aggregating Unstructured Crowd Data.

There is increasing interest in tapping the wisdom of the crowd for a wide variety of information needs. Our focus is on extracting reliable aggregate information from a set of unstructured textual inputs provided by the crowd. For example, given a set of user comments on an article, identify the fraction of supporters of the articles. Existing method are based on aggregating per-instance predictions from a classifier. Given the inherent inaccuracy of learning models, the question we ask is: can we obtain more accurate summaries through alternative paradigms? We have developed a method based on Kernel Mean Matching that provides more accurate estimates, that are also very efficient to deploy in practice.

Estimating classifier accuracy.

A practical problem facing many industrial deployments of imprecise prediction models is calibrating the accuracy of the model on large unlabeled data. Let $C(\mathbf{x})$ denote a model and D the unlabeled dataset. A conventional method of evaluating the accuracy of $C(\mathbf{x})$ on D is to manually select a labeled set L out of D, invoke $C(\mathbf{x})$ on each instance in L and aggregate the individual errors to calculate the accuracy of $C(\mathbf{x})$. When D is large in comparison to L, the accuracy measured this way is unlikely to be a reliable estimate of the classifier's real performance on the deployment data D. We propose [30] a method based on stratified sampling for estimating accuracy and select instances for labeling in a loop. For stratifying data we develop a novel strategy of learning r bit hash functions to preserve similarity in accuracy values.

Top-K count queries over duplicates.

Suppose we are interested in finding the K most frequently mentioned entities in a dataset containing many noisily duplicated entities. We show how to dedup on the fly only the part of the data actually needed for the answer — a requirement in massive and rapidly evolving sources where batch deduplication is not feasible. We propose a novel method of successively collapsing and pruning records which yield an order of magnitude reduction in running time compared to deduplicating the entire data first. We also show how to return multiple high scoring answers to handle situations where it is impossible to resolve if two records are indeed duplicates of each other [43].

8. STATISTICAL RELATIONAL LEARNING

The emerging area of statistical relational learning (SRL) is characterised by a number of distinct strands of research. Especially prominent is research concerned with the construction of parametric and non-parametric models from data that consist of multiple relations. To a first approximation, this is the kind of data that can be stored in multiple tables of a relational database, although it can get more complicated than this. Interest in this form of modelling is driven by at least two different trends: (1) The data are no longer simply values of known (pre-defined) features, but are in the form of observations of several inter-related variables. In such situations, it is often impractical to pre-define all kinds of features that may be of interest; and human expertise may not be available to define new sets of features for each new situation and modelling task. (2) Data is now available in very large quantities, largely due to advances in automation and the low-cost of secondary storage.

Research in SRL is concerned principally with different ways of representing the relational information; techniques of combining these representations with the calculus of probability; estimation of parameters of distributions and inference to yield probabilities with model predictions. The field is currently at an early stage, and shows great potential for the modelling of complex systems.

In the older, but related, field of Inductive Logic Programming (ILP), there has been substantial work of an engineering flavour specifically aimed at combining relational and statistical modelling that may be directly applicable to the kinds of data described here. This research consists of the use of ILP systems—programs that are capable of learning relations in first-order logic— as a tool for discovering useful features for subsequent use by any standard statistical model.The case for this form of data analysis is that the discovery of relational features must necessarily require some form of first-order learning, of which ILP systems are an instance.

Arguments in-principle aside, there are several reports in the literature that augmenting any existing features with ILP-discovered relational features can substantially improve the predictive power of a statistical model. While this approach is simple, and there appears to be experimental evidence to suggest it is effective, much still needs to be done to scale these up to meet modern data and model requirements. This includes the abilities to discover features using very large datasets stored in secondary memory; from relational data arriving in a streaming manner; and from data which do not conform easily to expected patterns. The SRL research at IIT Bombay is concerned with scaling-up relational feature-discovery methods to an industrial strength. Specifically, research undertaken are in the following areas:

Conceptual. The purpose here is to investigate conceptual ways in which relational structures can be discovered efficiently and effectively from extremely large search spaces. By this we mean: constraints that can be imposed on features and structures without serious loss of expressive power; transformations of the feature-discovery problem to other tasks for which efficient algorithms are known; optimisation formulations that can be solved efficiently, learning and inferencing with structured output spaces and so on. We have pursued the following strategies: (a) pose the problem as a discrete optimisation problem and solve it heuristically [28, 15, 48, 39, 49], (b) pose the problem as a continuous (often convex) optimisation problem with sparsity inducing regularizers and solves it optimally [27, 36] and (c) study restrictions on the space of relational features and investigate empirically whether it is acceptable for a relational learner to examine a more restricted space of features than that actually necessary for the full statistical model [41, 35, 33] We have also looked at heuristics for speeding up inference algorithms in relational settings [34].

Application. The purpose here is to investigate the applicability of feature-based techniques to data analysis tasks of constructing discriminatory and generative models for problems such as information extraction and disambiguation [33, 15, 48, 39, 49, 50].

9. EXECUTING CONTINUOUS QUERIES OVER DATA AGGREGATORS

Queries on web data, very often, involve distributed data sources. If the data items are dynamic, query results need to be refreshed continuously to avoid the risk of results becoming stale. We have developed techniques for executing continuous aggregation queries over data aggregators with the minimum number of refresh messages between the data sources [19], the aggregators and the client [20], leading to significant improvement in the utilization of network and computational resources.

In continuous query applications, usually the user is not interested in all the updates: a user either specifies an incoherency bound, where the user can tolerate some bounded inaccuracy in the query results; or a selection condition (e.g., threshold), such that the user is interested only if the query value satisfies the condition. Data aggregators can pull the data values from data sources or the data sources can push the values to an aggregator. Different data aggregators can execute either different sets of queries; or divide the queries such that different aggregators execute sub-queries of the individual queries [45]. Further, the aggregation functions used in the query can either be linear functions like SUM and AVG, or non-linear functions such as ratio, MAX, non-linear polynomial [21], etc. Although there exist various point solutions to the problem, as far as we know, our work is the first to cover all the dimensions [23].

As an example of the cases where threshold values are used to limit the number of refresh messages, consider ratio threshold queries (RTQs) where the user is interested in knowing whether the ratio of two aggregations has crossed a user specified threshold [22]. Such queries are executed by assigning conditions for individual data sources so that the data sources refresh the data values only if the assigned conditions are violated. We assign these source conditions such that no violation of the client threshold condition is missed while minimizing the number of message exchanges between the data sources and the data aggregator. Using performance evaluation we have shown that our method of assigning the source conditions results in up to an order of magnitude fewer refresh messages compared to the methods proposed in literature.

10. EXECUTING CATEGORY BASED QUERIES OVER DYNAMIC DATA

In order to harness the information present in unstructured dynamic data such as blog posts, forum postings, etc. we have been developing specialized techniques which given a keyword query, find the top-K categories related to the query. Example categories returned by such a category based search system for a keyword query say '911' would include: Information about September 11 attacks, Information about emergency services in US, Information about cars (Porsche 911), Information about 900 AD, etc. Thus category based search systems are able to provide a macroscopic view of the information present in the data [7].

There are multiple dimensions to the problem. The first is the nature of the query execution. In certain domains where users want to track the evolving information present in the dynamic data, users would want to be updated continuously of the changes in the top-K categories for their keyword queries. For such domains, we continuously report the top-K categories for the user query [9, 10]. On the other hand, in domains where the user wants to perform 'data exploration' a user would be satisfied with point queries where the top-K categories are reported only at the time when the query is executed [8]. Another dimension is the nature of the accuracy requirement. Users may not be interested in finding the exact answers to their top-K keyword queries but may be satisfied with say 80% accurate results. In others, it might be imperative that the system provides answers without any infidelity. Based on the above dimensions, we have built and evaluated systems that tackle the following types of queries over dynamic data: Category Based Exact Point Queries, Category Based Exact Continuous Queries, Category Based Incoherency Bounded Continuous Queries and Category Based Incoherency Bounded Point Queries. This work is one of the first to explore the problems associated with executing category based queries over dynamic data. We have evaluated them using real world data which show the superior performance of our techniques as compared to known alternatives.

11. REAL TIME TOPIC DETECTION OVER UNSTRUCTURED DATA STREAMS

Existing techniques for discovering emerging topics from a microblog stream in real time (such as Twitter trending topics), have several lacunae; extant graph based event detection techniques are not practical in microblog settings due to their complexity; and conventional techniques, which have been developed for blogs, webpages, etc., involving the use of keyword search, are only useful for finding information about known events. Our techniques discover topics that are unraveling in microblog message streams in real time so that such topics can be reported as soon as they occur[3].

Let S_i represent a set of keywords (potentially spread over multiple messages) from a unique user *i* in a time window that spans from time $(t - \delta.w)$ to current time *t*, where δ represents unit time called *quantum* and $\delta.w$ is the length of the time window. Let S_w^t={$S_1 \ldots S_m$} be a set of keywords sent by *m* unique users in the microblog stream in a given time window. S_w^t contains the messages from a sliding window (of size $\delta.w$) over the message stream. Time unit δ denotes the fixed rate at which the window is moved forward. We represent all the keywords, after removing stop words, appearing in the messages in the current window as nodes in an undirected graph Correlated Keyword Graph (CKG). CKG is a dynamic graph whose state at time *t*, is $G^t = (V^t, E^t)$ where V^t is the subset of keywords appearing in message set S_w^t. Thus, two keywords are said to be temporally correlated *iff* they appear in V^t and are said to be spatially correlated if they have an edge between them in E^t. An edge links two keywords *iff* they both appear in a keyword set S_i belonging to a user *i*.

Hence, in [3], we model the problem as discovering dense clusters in highly dynamic graph CKG. Half-quasi cliques are considered as clusters of interest as this leads to good precision and recall of discovered events. In order to find clusters, we propose and exploit a novel graph property which we call *short-cycle property*. Further we present a novel ranking function to identify the important events. We show that globally consistent ranking of events can be achieved by exploiting local properties of clusters.

12. HOLISTIC QUERY OPTIMIZATION

Queries, or calls to stored procedures/user-defined functions are often invoked multiple times, either from within a loop in an application program, or from the where/select clause of an outer query. When the invoked query or function involves database access, a naive implementation can result in very poor performance, due to random I/O. Query decorrelation addresses this problem in the special case of nested sub-queries, but is not applicable otherwise. This problem is traditionally addressed by manually rewriting the application to make it set-oriented, by creating a batch of parameters, and by rewriting the query/procedure to work on the batch instead of one parameter at a time. Such manual rewriting is time-consuming and error prone.

We have been working on an approach which we call holistic query optimization, which combines program analysis and rewriting of imperative application programs, with the rewriting of database queries, with the goal of optimizing the database accesses of an application. Our early work in this area (Guravannavar and Sudarshan [26]) introduced a technique for program analysis and rewriting to automatically replace queries inside loops by batched versions of the queries. In [18], we explored how to achieve similar effects using asynchronous query submission, which is useful when batching cannot be done, e.g. for Web service calls. We have built a tool called DBridge for holistic optimization of Java programs using the JDBC framework [17]. In Ramachandra and Sudarshan [38] we explored to how introduce query result prefetching into complex procedure call sequences, with minimal program rewriting. Such prefetching could be very useful for optimizing programs written using Object-Relational Mapping frameworks such as Hibernate. Performance studies of our system show significant reduction in cost due to our techniques.

13. XDATA: GENERATING TEST DATA TO KILL SQL QUERY MUTATIONS

Complex SQL queries are widely used today, but it is difficult to check if a complex query has been written correctly. Formal verification based on comparing a specification with an implementation is not applicable, since SQL queries are essentially a specification without any implementation. Queries are usually checked by running them on sample datasets and checking that the correct result is returned; there is no guarantee that all possible errors are detected.

In this project, we address the problem of test data generation for checking correctness of SQL queries, based on the query mutation approach for modeling errors. In this approach, errors are modeled as small changes ("mutations") to an intended query, and the goal is to generate test data that can distinguish between the original query and the mutant, i.e, kill the mutant, In [46], we focus on test data generation to kill a particular on a class of join/outer-join mutations, comparison operator mutations, and aggregation operation mutations, which are a common cause of error. To minimize human effort in testing, our techniques generate a test suite containing small and intuitive test datasets. The number of datasets generated, is linear in the size of the query, although the number of mutations in the class we consider is exponential. Under certain assumptions on constraints and query constructs, the test suite we generate is complete for a subclass of mutations that we define, i.e., it kills all non-equivalent mutations in this subclass. We have subsequently extended this work to handle more SQL constructs and types of mutations; we note that completeness in terms of killing mutants is not guaranteed for some of the extensions, but the tool is useful for testing, even without guaranteeing completeness.

We are currently building a tool to help in grading student SQL assignments [16] which works as follows. The tool first generates test datasets from a correct sample answer to an SQL assignment; for each student query, the tool compares the result of the student query with the result of the correct query, on each of these datasets. A mismatch on any of the dataset is evidence of an error in the student query. We ran a large number of actual student SQL queries through our system, and found that our system outperformed both TAs who had corrected the assignments manually, and two sample datasets which had been used to check for correctness of the queries. We believe such a tool could be of great value to instructors of database courses, and plan to open-source the tool.

14. CONTRIBUTIONS TO THE COMMUNITY

In addition to the above research-oriented activities, our efforts have resulted in several artifacts, of potential benefit to the community at large:

- S. Sarawagi's software for segmenting and cleaning Indian addresses, deployed in banks and other financial institutions.
- Conditional Random Fields based information extraction package, developed by S. Sarawagi and placed in open source[1].
- Source code for S. Sudarshan's PYRO multi-query optimizer and the BANKS system for keyword search are available on request.
- In the spirit of Weka, G. Ramakrishnan has developed a relational learning (RL) workbench called BET (Background + Examples = Theories) implemented in Java[2]. The objective of BET is to shorten the learning curve of users (including novices) and to facilitate speedy development of new RL systems as well as quick integration of existing ILP systems.
- K. Ramamritham and his colleagues have developed and deployed *almost All QUestions Answered* (`aAQUA` [3]), a multilingual, multimedia agricultural portal that lets rural farmers and agribusiness employees ask and answer questions online. The system incorporates novel database systems and information retrieval techniques, including intelligent caching and offline access with intermittent synchronization.
- S. Sudarshan has coauthored a textbook on database systems, which is widely used all over the world [47].
- Chakrabarti's book, Mining the Web [11], published in 2002, continues to be among the standard texts used in graduate courses worldwide on Web crawling, indexing, search and hyperlink analysis. A second edition is in preparation.

15. REFERENCES

[1] A. Agarwal and S. Chakrabarti. Learning random walks to rank nodes in graphs. In *ICML*, 2007.

[2] A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entities. In *SIGKDD Conf.*, pages 14–23, 2006.

[3] M. K. Agarwal, K. Ramamritham, and M. Bhide. Real time discovery of dense clusters in highly dynamic graphs: Identifying real world events in highly dynamic environments. In *VLDB'12: Proc. of 38th Intl. Conf. on Very Large Data Bases*, 2012.

[1] `http://crf.sf.net`

[2] `http://www.cse.iitb.ac.in/~bet/`

[3] `http://www.aaqua.org`

[4] A. Agrawal, S. Sudarshan, A. Sahoo, A. Sandalwala, and P. Jaiswal. Entity ranking and relationship queries using an extended graph model. In *Intl. Conf. on Management of Data (COMAD)*, 2012.

[5] S. Banerjee, S. Chakrabarti, and G. Ramakrishnan. Learning to rank for quantity consensus queries. In *SIGIR Conf.*, 2009.

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

[7] M. .Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy. Adaptive push pull: Disseminating dynamic web data. *IEEE Transactions on Computers*, 51:652–668, 2002.

[8] M. Bhide, K.Ramamritham, and P.Roy. Keyword search over dynamic categorized information. In *Intl. Conf. on Data Engineering*, 2009.

[9] M. Bhide and K. Ramamritham. Category based infidelity bounded queries over unstructured data streams. In *IEEE Transactions on Knowledge and Data Engineering*, 2013.

[10] M. Bhide, K. Ramamritham, and M. Agrawal. Efficient execution of continuous incoherency bounded queries over multi-source streaming data. In *Intl. Conf. on Distributed Computing Systems*, 2007.

[11] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[12] S. Chakrabarti, S. Kasturi, B. Balakrishnan, G. Ramakrishnan, and R. Saraf. Compressed data structures for annotated web search. In *WWW Conf.*, pages 121–130, 2012.

[13] S. Chakrabarti, A. Pathak, and M. Gupta. Index design and query processing for graph conductance search. *VLDB Journal*, 2010.

[14] S. Chakrabarti, D. Sane, and G. Ramakrishnan. Web-scale entity-relation search architecture (poster). In *WWW Conf.*, pages 21–22, 2011.

[15] A. Chalamalla, S. Negi, L. V. Subramaniam, and G. Ramakrishnan. Identification of class specific discourse patterns. In *CIKM*, pages 1193–1202, 2008.

[16] B. Chandra, B. Chawda, S. Shah, and S. Sudarshan. Extending XData to kill SQL query mutants in the wild. Unpublished manuscript, 2012.

[17] M. Chavan, R. Guravannavar, K. Ramachandra, and S. Sudarshan. DBridge: A program rewrite tool for set-oriented query execution (demo). In *ICDE*, pages 1284–1287, 2011.

[18] M. Chavan, R. Guravannavar, K. Ramachandra, and S. Sudarshan. Program transformations for asynchronous query submission. In *ICDE*, pages 375–386, 2011.

[19] R. Gupta, A. Puri, and K. Ramamritham. Executing incoherency bounded continuous queries at web data aggregators. In *WWW '05: Proc. of the 16th Intl. Conf. on World Wide Web*, Chiba, Japan, 2005.

[20] R. Gupta and K. Ramamritham. Optimized query planning of continuous aggregation queries in dynamic data dissemination networks. In *WWW '07: Proc. of the 16th Intl. Conf. on World Wide Web*, pages 321–330, Banff, Alberta, Canada, 2007.

[21] R. Gupta and K. Ramamritham. Optimized query planning of continuous aggregation queries over a network of data aggregators. In *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[22] R. Gupta, K. Ramamritham, and M. Mohania. Executing ratio threshold queries over distributed data sources. In *ICDE '10: Proc. of 26th IEEE Intl. Conf. on Data Engineering*, 2010.

[23] R. Gupta, K. Ramamritham. Scalable Execution of Continuous Aggregation Queries over Web Data. In *IEEE Internet Computing* 16(1): 43-51 2012.

[24] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. In *PVLDB*, 2009.

[25] R. Gupta and S. Sarawagi. Joint training for open-domain extraction on the web: Exploiting overlap when supervision is limited. In *WSDM*, 2011.

[26] R. Guravannavar and S. Sudarshan. Rewriting procedures for batched bindings. *PVLDB*, 1(1):1107–1123, 2008.

[27] P. Jawanpuria, J. S. Nath, and G. Ramakrishnan. Efficient rule ensemble learning using hierarchical kernels. In *ICML*, pages 161–168, 2011.

[28] S. Joshi, G. Ramakrishnan, and A. Srinivasan. Feature construction using theory-guided sampling and randomised search. In *ILP*, pages 140–157, 2008.

[29] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *VLDB*, 2005.

[30] N. Katariya, A. Iyer, and S. Sarawagi. Active evaluation of classifiers on large datasets. In *ICDM (***Runner-up for Best paper award***)*, 2012.

[31] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in Web text. In *SIGKDD Conf.*, pages 457–466, 2009.

[32] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. In *VLDB*, 2010.

[33] A. Nagesh, G. Ramakrishnan, L. Chiticariu, R. Krishnamurthy, A. Dharkar, and P. Bhattacharyya. Towards efficient named-entity rule induction for customizability. In *EMNLP-CoNLL*, pages 128–138, 2012.

[34] N. Nair, A. Govindan, C. Jayaraman, K. TVS, and G. Ramakrishnan. Pruning search space for weighted first order horn clause satisfiability. In *ILP*, 2010.

[35] N. Nair, A. Nagesh, and G. Ramakrishnan. Probing the space of optimal markov logic networks for sequence labeling. In *ILP*, 2012.

[36] N. Nair, A. Saha, G. Ramakrishnan, and S. Krishnaswamy. Rule ensemble learning using hierarchical kernels in structured output spaces. In *AAAI*, 2012.

[37] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. In *Proc. of the 36th Int'l Conf. on Very Large Databases (VLDB)*, 2012.

[38] K. Ramachandra and S. Sudarshan. Holistic optimization by prefetching query results. In *SIGMOD*, pages 133–144, 2012.

[39] G. Ramakrishnan, S. Joshi, S. Balakrishnan, and A. Srinivasan. Using ilp to construct features for information extraction from semi-structured text. In *ILP*, pages 211–224, 2007.

[40] A. Ramesh, S. Sudarshan, P. Joshi, and M. N. Gaonkar. Keyword search on form results. *VLDB J.*, 22(1):99–123, 2013.

[41] A. Saha, A. Srinivasan, and G. Ramakrishnan. What kinds of relational features are useful for statistical learning? In *ILP*, 2012.

[42] S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPS*, 2004.

[43] S. Sarawagi, V. S. Deshpande, and S. Kasliwal. Efficient top-k count queries over imprecise duplicates. In *EDBT*, 2009.

[44] U. Sawant and S. Chakrabarti. Learning joint query interpretation and response ranking. In *WWW Conf.*, Brazil, 2013.

[45] S. Shah, K. Ramamritham, and P. J. Shenoy. Maintaining coherency of dynamic data in cooperating repositories. In *VLDB*, pages 526–537. Morgan Kaufmann, 2002.

[46] S. Shah, S. Sudarshan, S. Kajbaje, S. Patidar, B. P. Gupta, and D. Vira. Generating test data for killing SQL mutants: A constraint-based approach. In *ICDE*, pages 1175–1186, 2011.

[47] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, 6th edition, 2010.

[48] L. Specia, A. Srinivasan, S. Joshi, G. Ramakrishnan, and M. das Graças Volpe Nunes. An investigation into feature construction to assist word sense disambiguation. *Machine Learning*, 76(1):109–136, 2009.

[49] L. Specia, A. Srinivasan, G. Ramakrishnan, and M. das Graças Volpe Nunes. Word sense disambiguation using inductive logic programming. In *ILP*, pages 409–423, 2006.

[50] A. Srinivasan and G. Ramakrishnan. Parameter screening and optimisation for ilp using designed experiments. *Journal of Machine Learning Research*, 12:627–662, 2011.