The Future of DB & IR

Gerhard Weikum, Gjergji Kasneci, Maya Ramanath, Fabian Suchanek
Max Planck Institute for Informatics
D-66123 Saarbruecken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

This article advocates that database-systems (DB) and information-retrieval (IR) methods be integrated to address the needs of important applications that emerge with the ongoing explosion and diversification of digital information. A grand challenge that a joint DB&IR endeavor could aim for is to automatically build and maintain a comprehensive knowledge base that encompasses all important facts from encyclopedic sources and the scientific literature. It should represent facts in terms of typed entities and relations, and allow expressive queries that return ranked results with high precision in an efficient and scalable manner. The article presents various approaches and the roles of DB and IR methods towards this ambitious objective.

1. TWO PARALLEL UNIVERSES?

Database systems (DB) and information retrieval (IR) are two separate fields of computer science by historical accident. Both study concepts, models, and computational methods for managing large amounts of complex information, but thirty or forty years ago they started with very different application areas as major motivations and technology drivers: accounting systems (online reservations, banking, etc.) for DB, and library systems (bibliographic catalogs, patent collections, etc.) for IR. Thus, the two directions and their research communities emphasized very different aspects of information management: data consistency, precise query processing, and efficiency on the DB side, and text understanding, statistical ranking models, and user satisfaction on the IR side.

Decades later, there is now growing awareness of the needs for integrating DB and IR technologies. There have already been attempts at addressing this integration ten years ago, most notably, the probabilistic datalog and probabilistic relational-algebra models [13, 14], the proximal node model [19], and the WHIRL approach to similarity joins [9]. But it is only in recent years that mission-critical applications have been emerging with strong desire for integrated DB&IR methods and platforms. From an IR viewpoint, digital libraries of all kinds are becoming very rich information repositories with documents augmented by metadata and annotations captured in semistructured data formats like XML; enterprise search on intranet data can be seen as a specific variant of this theme. From a DB viewpoint, application areas such as customer support, product and market research, or health-care management exhibit tremendous data growth, in terms of both structured and unstructured information. Web 2.0 applications like social communities require support for structured and textual data as well as ranking and recommendation in the presence of uncertain information of highly diverse quality.

The co-existence of DB and IR is illustrated in the leftmost quad chart of Figure 1. Here, information systems are categorized by two dimensions: the nature of the data to be managed and the nature of how the data is searched. The first dimension divides the digital world into structured data like schema-oriented records with numerical, categorical, and short-string attributes vs. unstructured data like natural-language text and multimodal information (speech, video, etc.) or loose collections of heterogeneous records. The second dimension discriminates sophisticated query languages that can express logical conditions vs. simple keyword search as the prevalent way of posing queries to search engines. For several decades it has appeared that DB and IR systems reside in two disjoint quadrants of the chart, and that the other two quadrants are useless or at least unoccupied.

In the last decade, both DB and IR researchers have been exploring the previously blank quadrants of the chart, as shown in the middle of Figure 1. IR-style keyword search over structured data like relational databases makes sense whenever the structural data description – the schema – is so complex that information needs cannot be concisely or conveniently expressed in a structured query. For example, consider a social-network database with tables about users, friends, posted items (such as photos, videos, or recommended books or songs), ratings, comments, etc. Assume that a user wants to find what connections Alon, Raghu, and Surajit share with respect to the Semantic Web. Interesting answers may include that all three co-authored a book on the Semantic Web, or two of them edited a book and the third one commented on it, or they are all friends and one of them posted a video entitled "Semantic Web Saga". With structured querying, where each value (e.g., "Alon") refers to a particular attribute (e.g., User.Name or Friend.Name), the combinatiorial options lead to very complex queries with many joins and unions. It is much simpler to merely state five keywords "Alon, Raghu, Surajit, Semantic, Web", and let the system compute the most meaningful answers in a relational graph. This relaxed attitude towards the schema

(which value should occur in which attribute) naturally entails IR-style ranking.

Conversely, linguistic and learning-based information-extraction techniques have been applied in order to augment textual sources with structured records and enable expressive DB-style querying over originally unstructured data. Consider an information request about *the life of the scientist Max Planck*, to be evaluated over an XML-based digital library, say an extended form of Wikipedia. A simple approach would be to formulate a keyword query like *"life scientist Max Planck"*. Unfortunately, the results would be dominated by information about Max-Planck Institutes (there are about 80 such institutes in Germany), for example, in the area of life sciences. Structured query languages like SQL or XPath Full-Text allow us to specify more precisely what we are interested in, for example, in the form of attribute name-value conditions such as *Name = "Max Planck"* or XML structure-and-content conditions such as
//Article [Person ftcontains "Max Planck"]
[Category ftcontains "science"] //Biography.
These search predicates yield much more precise answers, but may now require approximate matching (to counter over-specified queries) and result ranking (see, e.g., [2, 25] and references given there).

A final observation is that the initially pure quadrants for DB and IR systems have been substantially enhanced by new methods for digital libraries, enterprise search and analytics, text extensions for database engines, ranking capabilities for SQL and XQuery, and many more. The boundaries between quadrants are getting blurred; the DB and IR fields are fertilizing each other. The rightmost chart of Figure 1 envisions convergence towards an integrated solution; the future will show if and how this goal is feasible.

2. THE NEED FOR DB&IR

Many applications need to manage both structured and unstructured data. Consider a scenario in the medical healthcare area where we may have relational tables with the following schemas (i.e., attribute names and types, unique keys being underlined):

Disease	(*DId int*, *Name char*[50], *Category int*, *Pathogen char*[50], ...)
Patient	(*PId int*, ..., *Age int*, *TreatedDId int*, *ResponsibleHId int*, *Timestamp date*, *Report longtext*, ...)
Hospital	(*HId int*, *Address char*[200], ...)

While some of the information, especially foreign-key references between relations (e.g., a patient record referring to a disease identifier), is suitable for structured queries, long text fields that often contain valuable latent information are only amenable to keyword and text-similarity search. Moreover, some of the attributes like Category may refer to external taxonomies and ontologies such as UMLS (Unified Medical Language System).

Now consider an information request such as: *Find young patients in central Europe who have been reported, in the last two weeks, to have symptoms of tropical virus diseases and an indication of anomalies.* Computing relevant answers requires evaluating structured predicates like range conditions on Age or joins with additional ontology tables for identifying those values of disease Name, Category, and Pathogen that refer to tropical virus diseases. It also involves fuzzy predicates on Report to test the "anomaly" condition. Moreoever, hospital Address needs to be matched against geographic taxonomies with some inherent vagueness (e.g., about England or Italy being considered central Europe or not). Finally, with such fuzzy matchings and similarity tests it is absolutely necessary to provide meaningful rankings of the results. For example, a case with strong evidence of anomalies that may be at the border of central Europe or slightly outside or happened slightly outside the two-week time window should rank higher than a case that satisfies the structured conditions very well (including the spatio-temporal conditions) but has little evidence in the Report that it was particularly anomalous.

So structured and unstructured search conditions are combined into a single query, and the query results need to be ranked. Such queries must be efficiently evaluated over very large datasets that exhibit high update rates as new cases of diseases are added. Clearly, one can build such an application by using two separate platforms, a DB system for the structured data and an IR search engine for the textual and fuzzy-matching issues. But this, still widely adopted, approach puts a high burden on the application developers as many tasks are not covered by the underlying platforms and thus need to be addressed in the application code. An integrated DB&IR platform would greatly simplify the development of the application and would largely reduce the cost of maintaining it and adapting it to future needs.

Abstracting from this kind of application-centric discussion, there are compelling motivations for bringing IR concepts to DB systems and vice versa, leading to the following DB&IR desiderata:

- *Approximate matching and record linkage:* Adding text-matching functionality to DB systems often entails approximate matching (e.g., because of spelling variants) and, when text fields refer to named entities, leads into record linkage for matching entities. For example, the strings "William J. Clinton" and "Bill Clinton" likely denote the same person, and the names "M-31" and "NGC 224" should be reconciled to denote the Andromeda galaxy. Approximate matching by similarity measures requires IR-style ranking.
- *Too-many-answers ranking:* Preference search, for example, over travel portals or product catalogs, often poses a too-many-answers problem. Narrowing the query conditions may overshoot by producing too few or even no results; interactive reformulation and browsing is time-consuming and may irritate users. Large result sets inevitably require ranking, based on data and/or workload statistics as well as user profiles.
- *Schema relaxation and heterogeneity:* In the DB world, it has become the norm that applications access multiple databases, often with a run-time choice of the data sources. Even if each of these sources contains structured data records and comes with an explicit schema, there is no unified global schema unless a breakthrough were achieved to magically perform perfect on-the-fly data integration. So the application program has to cope with the heterogeneity of the underlying schema names, XML tags, or RDF properties, and queries need to be schema-agnostic or tolerant to schema relaxation.

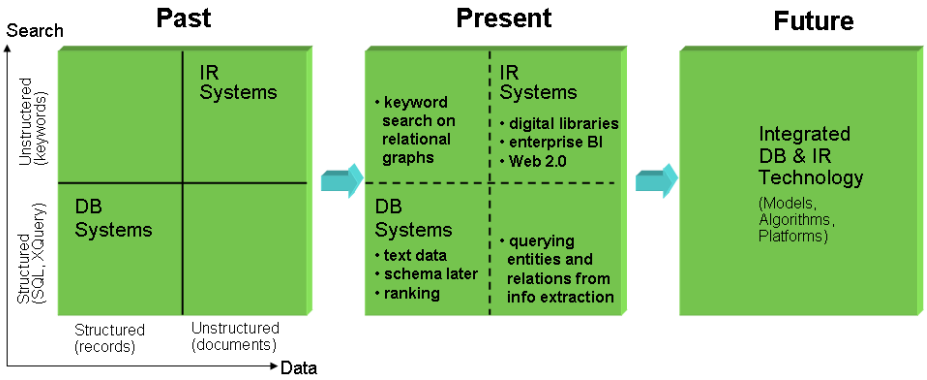

Figure 1: Past, Present, and Future of DB&IR

- *Information extraction and uncertain data:* Textual information contains named entities and relationships in natural-language sentences. These can be made explicit by information-extraction techniques (pattern matching, statistical learning, natural-language processing). This can potentially lead to large knowledge bases whose facts, however, exhibit some uncertainty. Querying the extracted facts thus entails the need for ranking.

- *Entity search and ranking:* Recognizing entities in text sources allows entity-search queries about electronics products, travel destinations, movie stars, etc., thus boosting the search capabilities on intranets, portals, news feeds, and the business- and entertainment-oriented parts of the Web. Extracting binary relations between entities and also place and time attributes could pave the way towards semantic IR on digital libraries (e.g., PubMed), news, and blogs, and could also aid natural-language question answering and searching the Deep Web.

3. THE FUTURE: HARVESTING, SEARCHING, AND RANKING KNOWLEDGE FROM THE WEB

3.1 Opportunities

The Web bears the potential of being the world's most comprehensive knowledge base, but we are far from exploiting this potential. Valuable scientific and cultural content is all mixed up with huge amounts of noisy, low-quality, unstructured text and media. The challenge is: How can we extract the important facts from the Web and organize them into an explicit knowledge base that captures entities and semantic relations between them? Imagine a "Structured Wikipedia" that has the same scale and richness as Wikipedia itself but offers a precise and concise representation of knowledge that enables expressive and highly precise querying. Figure 2 illustrates what such a knowledge base may look like. The figure depicts an excerpt of the YAGO knowledge base [24] (further discussed below), which is a typed entity-relationship graph but can equivalently be represented in the RDF or Owl-Lite data models. Building and maintaining this knowledge base in a largely automated manner is not only a difficult problem, but also an opportunity for computer science to contribute towards high-value assets for science, culture, and society. DB and IR methods could play major roles in this endeavor.

With a knowledge base that sublimates the valuable content from the Web, we could address difficult questions that are beyond the capabilities of today's keyword-based search engines. For example, one could ask for drugs that inhibit proteases and would obtain a precise and fairly comprehensive list of drugs for this HIV-relevant family of enzymes. Such advanced information requests are posed by knowledge workers like scientists, students, journalists, historians, or market researchers. Although it is possible today to find relevant answers, this process is extremely laborious and time-consuming as it often requires rephrasing queries and browsing through many potentially promising but eventually useless result pages. The following example questions illustrate this point:

- *Which German Nobel laureate survived both world wars and outlived all of his four children?* The answer is Max Planck. The bits and pieces for the answer are not that difficult to locate: lists of Nobel prize winners, birth and death dates of these people, family members extracted from biographies, dates of children. Gathering and connecting these facts is straightforward for a human, but it may take days of manually inspecting Web pages.

- *Which politicians are also accomplished scientists?* Today's search engines fail on questions of this kind because they are matching words and returning pages rather than identifying entities like persons and testing their relations. Additionally, the question entails a difficult ranking problem. Wikipedia alone contains hundreds of persons that are listed in the categories Politicians as well as Scientists. An insightful answer must rank important people first, for example, the German chancellor Angela Merkel who has a doctoral degree in physical chemistry, or Benjamin Franklin, and the like.

- *How are Max Planck, Angela Merkel, Jim Gray, and the Dalai Lama related?* The answer is that all four of them have a doctoral degree from a German university, honorary doctorates in the cases of Jim Gray and the Dalai Lama. Discovering interesting facts about multiple entities and their connections on the Web is

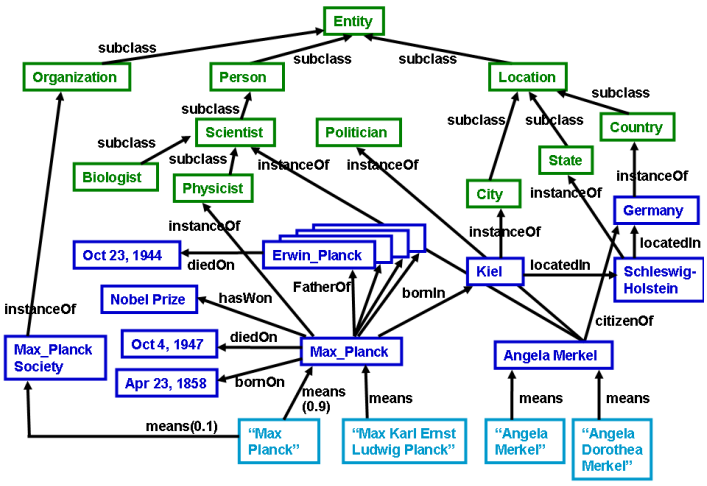

Figure 2: Excerpt of the YAGO Knowledge Base

virtually impossible, by the sheer amount of interconnected pages about the four people.

Note that even though the queries are phrased as natural-language questions, they would remain equally hard if they were expressed in a formal language. Conversely, a rich knowledge base of entities and relations would enable much more effective natural-language question answering.

3.2 Approaches

Information organization and search on the Web is gaining structure, context awareness, and more semantic flavor in the forms of faceted search, vertical-domain search, entity search, and Deep-Web search. All major search engines recognize a large fraction of product names, have built-in knowledge about geographic locations, and can return high-precision results for popular queries about consumer interests, traveling, and entertainment. Information-extraction and entity-search methods are clearly at work here. But these efforts seem to be focusing on specific domains only. Generalizing the approaches towards a universal methodology for knowledge harvesting requires bolder steps. Three major research avenues can contribute to this goal:

- *Semantic-Web*-style knowledge repositories like ontologies and taxonomies. These include general-purpose ontologies and thesauri such as SUMO, OpenCyc, or WordNet, as well as domain-specific ontologies and terminological taxonomies such as GeneOntology or UMLS in the biomedical domain.

- Large-scale information extraction (IE) from text sources in the spirit of a *Statistical Web*. IE methods – entity recognition and learning relational patterns – have recently become much more scalable, and also less dependent on human supervision [1, 10, 21].

- Social tagging and Web 2.0 communities that constitute the *Social Web*. Human contributions are abundant in the form of semantically annotated Web pages, phrases in pages, images, or videos, together providing "wisdom of the crowds". There are endeavors such as Freebase to collect structured data records from human communities. Wikipedia is another example of the Social-Web paradigm. It includes semistructured data like infoboxes that can be lifted into explicit facts [4, 24, 27].

Research projects often combine elements of the semantic, statistical, and social approaches. We discuss several interesting projects (omitting other relevant work for lack of space) and highlight results of our own project YAGO.

Libra: For supporting entity search on the Web, the Microsoft Research Lab in Beijing has developed comprehensive technology for information extraction. These include pattern-matching algorithms that are tailored to typical Web-page layouts, and also trained learning of patterns using advanced models like Hierarchical Conditional Random Fields [28]. A particular focus has been to extract entities and their attributes from product-related pages with HTML tables and lists. These methods and tools were used to build and maintain several vertical-domain portals, including product search and the *Libra* portal for scholarly search on the extracted records about authors, papers, conferences, and communities.

Once the facts are gathered and organized into a searchable form, a typical IR issue arises about how to rank the results of an entity-centric query. To this end, an advanced statistical language model (LM) has been extended from the level of document-oriented bags-of-words to structured records [20].

Libra is an example of the Statistical-Web approach.

Cimple / DBLife: The Cimple project [11, 22] is jointly carried out by the University of Wisconsin and Yahoo! Research. Similar to Libra, it aims to generate and maintain community-specific portals with structured information gathered from Web sources. However, it applies different methods to achieve this goal. This is illustrated by discussing Cimple's flagship application, the DBLife portal (`http://dblife.cs.wisc.edu`).

DBLife features automatically compiled "super-homepages" of researchers with bibliographic data as well as facts about

community services (PC work, etc.), colloquium lectures, and more. For gathering and reconciling these facts, Cimple has a suite of DB-style extractors based on pattern matching and dictionary lookups. These extractors are combined into execution plans, and periodically applied to a carefully selected set of relevant Web sources. The latter include prominent sites like DBLP and the Dbworld archive, and also important conference and university pages that are selected semi-automatically. While the overall approach makes use of IR concepts like $tf*idf$-based ranking and also Web-graph link analysis, the emphasis of Cimple is on a more DB-oriented toolkit for declarative extraction programs, using Datalog as a query-language framework and applying DB rewriting techniques for query optimization (cf. also [17]).

Cimple leans more towards the Semantic-Web rationale and less towards a Statistical-Web approach. In addition, it contains Social-Web elements, most notably, a Wiki-based mechanism for users to provide feedback about incorrect facts that they see on a community portal.

KnowItAll / TextRunner: Both Libra and Cimple operate on a per-page basis and then aim to extract as many facts as possible from the given page. A dual view is to focus on one or more entity types or relation types, and aim to populate these relations by inspecting many pages and exploiting their redundancies. For example, one may want to find all cities on this planet, all scientists, all guitar players, and other unary relations (entity types). For binary relations, consider gathering all CEO's of all companies, all (city, river) pairs where a city is located on a river, or answering questions like: who discovered what, or which enzyme triggers which biochemical process.

The KnowItAll project at the University of Washington in Seattle [5, 6, 12] has pursued this goal, using different techniques that combine pattern matching, linguistic analysis, and statistical learning. KnowItAll [12] starts with a set of *seeds*: instances of the relation of interest (e.g., a set of cities or a set of (city, river) pairs). This is the only "training input" that KnowItAll needs. The system can automatically find sentences on the Web that contain seeds, extract linguistic patterns surrounding seeds, perform statistical analyses to identify strong patterns, and finally lift the best patterns to obtain extraction rules. For example, the phrase templates *"located in downtown* x*"* and *"*x *is located on the banks of* y*"* may be determined to be good rules for extracting cities and (city, river) pairs, respectively. Now these rules can be applied to newly seen Web pages, yielding facts or fact candidates, some of which may in turn be considered as new, additional seeds. For identifying good rules and for assessing the confidence in the harvested facts statistical inference is needed.

The more recent TextRunner tool [5] has paid particular attention to scalability issues and has simplified the entire fact-gathering pipeline. It has a completely unsupervised bootstrapping phase for identifying simple patterns, just enough to identify, with high confidence, noun phrases and verbal patterns. When TextRunner sees a new Web page, it aggressively extracts all potentially meaningful instances of all possible binary relation types from the page text; [5] refers to this processing mode as *open information extraction* or "machine reading".

KnowItAll and TextRunner are examples of Statistical-Web methods for large-scale knowledge acquisition.

YAGO: Large-Scale Semantic Knowledge

Our own project YAGO (Yet Another Great Ontology) [23, 24] shares the goal of large-scale knowledge harvesting with KnowItAll and TextRunner, but in contrast, emphasizes high accuracy and consistency rather than high recall (coverage). YAGO is best characterized as a Semantic-Web approach.

YAGO primarily gathers its knowledge by integrating information from Wikipedia and WordNet. In addition, it can employ text-mining-based techniques as well. Currently, YAGO contains close to 2 million entities and about 20 million facts about them, where facts are instances of binary relations. Extensive sampling showed that the accuracy is at least 95 percent, and many of the remaining errors (false positives) are due to incorrect entries in Wikipedia itself. YAGO is publicly available at `http://www.mpi-inf.mpg.de/~suchanek/yago`.

Wikipedia provides two assets that can be seen as almost structured data: the infoboxes and the category system. Infoboxes are collections of attribute name-value pairs. They are often based on templates and reused for important types of entities such as countries, companies, scientists, music bands, sports teams, etc. For example, the infobox for Max Planck gives us data such as *birth_date = April 23, 1858*, *birth_place = Kiel*, *death_date = October 4, 1947*, *nationality = Germany*, or *alma_mater = Ludwig-Maximilians-Universität München*. As for the category system, the Max Planck article is manually placed into categories such as: *German_Nobel_laureates*, *Nobel_laureates_in_Physics*, *Quantum_physics*, or *University_of_Munich_alumni*. These give YAGO clues about instanceOf relations, and we can infer that the entity Max Planck is an instance of the classes GermanNobelLaureates, NobelLaureatesInPhysics, and UniversityOfMunichAlumni. But we have to be careful, as the placement in category *Quantum_physics* does not mean that Max Planck is an instance of QuantumPhysics. The YAGO extractors employ linguistic processing (noun phrase parsing) and mapping rules, to achieve high accuracy in harvesting the categories information.

The above examples indicate that solely relying on infoboxes and categories of Wikipedia may result in a large but incoherent collection of facts. For example, we may know that Max Planck is an instance of GermanNobelLaureates, but we may not be able to automatically infer that he is also an instance of Germans and an instance of NobelLaureates. Likewise, the fact that he is a physicist does not automatically tell us that he is a scientist. To address these shortcomings, YAGO makes intensive use of the WordNet thesaurus (light-weight ontology) and integrates the facts that are harvested from Wikipedia with the taxonomic backbone provided by WordNet.

While WordNet knows many abstract classes and both is-a and part-of relations among them, it has only sparse information about individual entities that would populate its classes. The wealth of entities in Wikipedia nicely complements WordNet; conversely, the rigor and high coverage of WordNet's taxonomy can make up for the gaps and noise in the Wikipedia category system. Each individual entity that YAGO discovers needs to be mapped into at least one of the existing YAGO classes. If this fails, the entity and its related facts are not admitted to the knowledge base. Analogously,

classes that are derived from Wikipedia category names such as GermanNobelLaureates need to be mapped, with a subclass relationship, to one or more superclasses such as NobelLaureates or Germans. These procedures ensure that we can maintain a consistent knowledge base, where consistency eliminates dangling entities or classes and guarantees that the subclass relation is acyclic.

Kylin / KOG: Recent work that also extracts information from Wikipedia is the "Intelligence in Wikipedia" project with its tools Kylin [26] and KOG [27]. Whenever an infobox type includes an attribute in some articles but this attribute has no value for a given article, Kylin analyzes the full text of the article to derive the most likely value. Like KnowItAll and TextRunner (and unlike Libra, Cimple, and YAGO), Kylin pursues open extraction by considering all potentially significant attributes even if they occur only sparsely in the entire Wikipedia corpus. KOG (Kylin Ontology Generator) builds on Kylin's output, unifies different attribute names, derives type signatures, and (like YAGO) maps the entities onto the WordNet taxonomy, using statistical relational learning [15]. KOG goes beyond YAGO by discovering new relation types. It builds on the class system of YAGO and DBpedia [4] (along with the entities in each class) to train its learning algorithms for generating the subsumption graph between classes.

This project uses a combination of all three paradigms: Semantic-Web-oriented by being targeted at infoboxes, Social-Web-based by leveraging the input of the large Wikipedia community, and Statistical-Web-style by employing learning methods.

Searching and Ranking YAGO with NAGA:

We have designed a query language for YAGO that adopts concepts from the standardized language SPARQL for RDF data, but extends these capabilities by more expressive pattern matching and provides ranking [18] (cf. also [3]). The prototype system that implements these features is coined NAGA (Not Another Google Answer); an online demo is accessible at `http://www.mpi-inf.mpg.de/~kasneci/naga`. As we can view the knowledge base as a graph, a query can be constructed by subgraph templates. Figure 3 shows three examples, related to our earlier question scenarios.

The leftmost query about politicans who are also scientists shows two nodes to be matched by the desired results and one node, labeled $\$x$, denoting a variable for which the query needs to find all bindings. The edge labels denote relations and need to be matched by the results. Here, *isa* is a shorthand notation for a composition of two connected edges that correspond to the relations *instanceOf* between an entity and a class and *subclass* between two classes. This way we also find people who belong to the classes *mayor* and *physicist*.

The query in the middle of Figure 3 (a simpler variant of the German-Nobel-laureate question) generalizes the point about labels referring to compositions of relations. The label *(bornIn|livesIn|citizenOf).locatedIn** is a regular expression that allows us to avoid overspecifying our information demand. We may be generous about when we call a person German, and the locatedIn relation often reflects geographical hierarchies, e.g., with cities, counties, states, and countries.

The rightmost query of Figure 3 is a broad relatedness query that looks for commonalities or other connections among several entities. Here again, we use regular expressions as edge labels in the graph template of the query.

NAGA queries often return too many results, so results need to be ranked. For example, a query like *What is known about Einstein?*, which may be phrased as a single-edge graph pattern $isa(Einstein, \$y)$, returns dozens if not hundreds of results including many uninteresting ones such as *isa(Einstein, Entity)*, *isa(Einstein, Organism)*, or *isa(Einstein, Colleague)*. Ranking models for such results are much more difficult than for traditional search engines, as we need to consider the graph structure in both queries and results. Three general criteria need to be accomodated:

- **Informativeness:** Users prefer informative answers: salient facts or interesting facts, as opposed to overly generic facts or facts that are trivially known already. In the example query about Einstein we would prefer the answers *isa(Einstein, Physicist)* or *isa(Einstein, NobelLaureate)* over the above near-trivial results or a non-trivial but still less important fact like *isa(Einstein, Vegetarian)*. However, when asking a different query about important vegetarians, the latter fact should well be one of the highest ranked results. Informativeness is a query-dependent (and potentially even user- and situation-dependent) notion.

- **Confidence:** We may occasionally have uncertain, dubious, or false statements in the knowledge base. Each fact is annotated with a confidence value derived from the fact's original data sources and the extraction methods that were used. The quality of the sources can be quantified in terms of authority and trust measures in the spirit of Google's PageRank, and the quality of different extractors can be empirically assessed. There are various ways of combining these measures into a single confidence value (see, e.g., [7, 8]), and high-confidence answers should be preferred.

- **Compactness:** Whenever a query returns paths or graphs rather than individual nodes, we are interested in compact graphs or short paths. For example, a query about how Einstein and Bohr are related should preferably return a short answer path like *both are physicists* rather than a convoluted answer like "Einstein was a vegetarian like Tom Cruise who was born in the same year in which Bohr died".

A good ranking function needs to combine all three criteria. We sketch our approach for the first one, informativeness. NAGA has developed a new kind of statistical language model (LM) for graph-structured data and queries. The parameters of the model are estimated from corpus or workload statistics. Consider the simple queries $isa(Einstein, \$y)$ about Einstein and $bornIn(\$y, Frankfurt)$ about people born in Frankfurt. For the Einstein query we estimate conditional co-occurrence probabilities $\frac{P[Einstein \wedge Physicist]}{P[Einstein]}$ and $\frac{P[Einstein \wedge Vegetarian]}{P[Einstein]}$ to compare and rank two possible answers. For the Frankfurt query we compute and compare $\frac{P[Goethe \wedge born \wedge Frankfurt]}{P[born \wedge Frankfurt]}$ against $\frac{P[Weikum \wedge born \wedge Frankfurt]}{P[born \wedge Frankfurt]}$, clearly favoring Goethe as a top result.

This LM-based ranking allows NAGA to rank politicians who are also scientists with high informativeness, with Benjamin Franklin, Angela Merkel, and other prominent figures showing up in the top ranks. So while the knowledge base

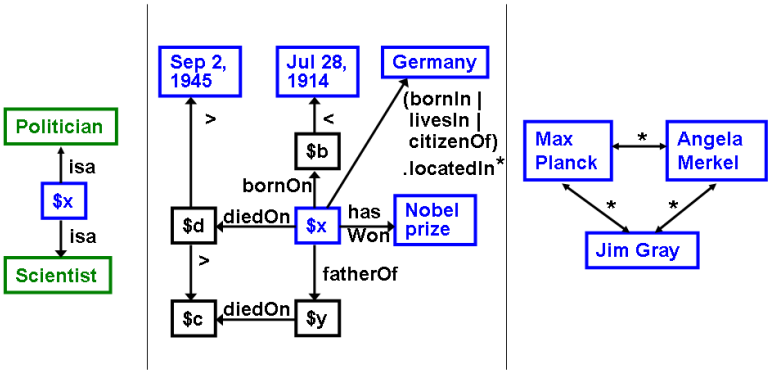

Figure 3: Example Queries for the YAGO Knowledge Base

YAGO is primarily a Semantic-Web endeavor, the ranking for its search engine very much builds on Statistical-Web assets.

3.3 Challenges

The outlined approaches have achieved good progress, but there are also major challenges ahead:

- **Scalable harvesting:** Most knowledge will be produced in textual form, e.g., in scientific publications. Methods for natural-language IE face inherent tradeoffs regarding training effort vs. easy deployment, and precision vs. recall of the results. Scaling up the IE machinery for higher throughput without sacrificing quality is a formidable problem. For example, can we process all blog postings on this planet at the same rate as they are produced, without missing relevant facts and without producing too many false positives?

- **Expressive ranking:** The LM-based ranking models pursued by Libra and NAGA should be extended to better capture the context of the user and the data. User context requires personalized and task-specific LM's that consider the current location, time, short-term history, and intentions in the user's digital traces. Data context calls for LM's for entity-relationship graphs, aiming to better model complex patterns beyond single facts (edges) and to consider types.

- **Efficient search:** Evaluating complex query predicates over graphs is computationally hard. Moreover, the need for ranking suggests that one should avoid materializing overly large numbers of results and better aim for solely computing the top-k results in a more efficient way (cf. [16]).

On a grander scale, the issue of the most appropriate paradigm arises. The three avenues Semantic, Statistical, and Social Web are by no means mutually exclusive. The outlined projects combine aspects of several of these directions. Deeper understanding of feedback between and synergies from the three paradigms is an overriding theme of great potential value. Semantic-Web sources can be powerful bootstrap tools for large-scale Statistical-Web mining. Statistical-Web tools may produce many false hypotheses, but these could be assessed by Social-Web platforms with large communities that engage in human-computing tasks. Social-Web endeavors in turn are often the grassroots catalysts for high-value knowledge repositories that eventually become Semantic-Web assets; Wikipedia and the derived knowledge bases such as YAGO exemplify this point.

4. CONCLUSION

This article has presented motivations for and approaches towards integrating the historically separated methodologies of the DB and IR fields. While deep DB&IR integration may still be wishful thinking for the time being, we do observe strong trends towards adopting IR concepts in the DB world and vice versa. In addition to applications that need to manage structured and unstructured data or highly heterogeneous information sources, one of the driving forces is the increasing interest and success in extracting entities and relations from text sources. The envisioned path towards automatically building and growing comprehensive knowledge bases with expressive search and ranking capabilities may take a long time to mature. In any case, it is an exciting challenge that should appeal to and benefit from several research communities, most notably, the DB and IR worlds.

5. ACKNOWLEDGMENTS

Our work on knowledge harvesting is supported by the Excellence Cluster "Multimodal Computing and Interaction" (`http://www.mmci.uni-saarland.de`) funded by the German Science Foundation (DFG).

6. REFERENCES

[1] Eugene Agichtein: Scaling Information Extraction to Large Document Collections. IEEE Data Eng. Bull. 28(4): 3-10 (2005)

[2] Sihem Amer-Yahia, Mounia Lalmas: XML Search: Languages, INEX and Scoring. SIGMOD Record 35(4): 16-23 (2006)

[3] Kemafor Anyanwu, Angela Maduko, Amit P. Sheth: SPARQ2L: Towards Support for Subgraph Extraction Queries in RDF Databases. WWW 2007: 797-806

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary G. Ives: DBpedia: A Nucleus for a Web of Open Data. ISWC/ASWC 2007: 722-735

[5] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni: Open

Information Extraction from the Web. IJCAI 2007: 2670-2676

[6] Michael J. Cafarella, Christopher Re, Dan Suciu, Oren Etzioni: Structured Querying of Web Text Data: A Technical Challenge. CIDR 2007: 225-234

[7] Soumen Chakrabarti: Dynamic Personalized PageRank in Entity-Relation Graphs. WWW 2007: 571-580

[8] Tao Cheng, Xifeng Yan, Kevin Chen-Chuan Chang: EntityRank: Searching Entities Directly and Holistically. VLDB 2007: 387-398

[9] William W. Cohen: Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. SIGMOD Conference 1998: 201-212

[10] Hamish Cunningham: An Introduction to Information Extraction. In Encyclopedia of Language and Linguistics, 2nd Edition, Elsevier, 2005

[11] Pedro DeRose, Warren Shen, Fei Chen, AnHai Doan, Raghu Ramakrishnan: Building Structured Web Community Portals: A Top-Down, Compositional, and Incremental Approach. VLDB 2007: 399-410

[12] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, Alexander Yates: Unsupervised Named-Entity Extraction from the Web: An Experimental Study. Artif. Intell. 165(1): 91-134, 2005

[13] Norbert Fuhr: Probabilistic Datalog - A Logic For Powerful Retrieval Methods. SIGIR 1995: 282-290

[14] Norbert Fuhr, Thomas Rölleke: A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM Trans. Inf. Syst. 15(1): 32-66, 1997.

[15] Lise Getoor, Ben Taskar (Editors): Introduction to Statistical Relational Learning. MIT Press, 2007

[16] Ihab F. Ilyas, George Beskales, Mohamed A. Soliman: A Survey of Top-k Query Processing Techniques in Relational Database Systems. ACM Computing Surveys, 2008

[17] Panagiotis G. Ipeirotis, Eugene Agichtein, Pranay Jain, Luis Gravano: Towards a Query Optimizer for Text-Centric Tasks. ACM Trans. Database Syst. 32(4): (2007)

[18] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, Gerhard Weikum: NAGA: Searching and Ranking Knowledge. ICDE 2008: 953-962

[19] Gonzalo Navarro, Ricardo A. Baeza-Yates: Proximal Nodes: A Model to Query Document Databases by Content and Structure. ACM Trans. Inf. Syst. 15(4): 400-435, 1997

[20] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, Wei-Ying Ma: Web Object Retrieval. WWW 2007: 81-90

[21] Sunita Sarawagi: Information Extraction. Foundations and Trends in Databases 2(1), 2008

[22] Warren Shen, AnHai Doan, Jeffrey F. Naughton, Raghu Ramakrishnan: Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. VLDB 2007: 1033-1044

[23] Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum: YAGO: a Core of Semantic Knowledge. WWW 2007: 697-706

[24] Fabian Suchanek, Gjergji Kasneci, Gerhard Weikum: YAGO: a Large Ontology from Wikipedia and WordNet. Journal of Web Semantics 6(3): 203-217, 2008

[25] Martin Theobald, Holger Bast, Debapriyo Majumdar, Ralf Schenkel, Gerhard Weikum: TopX: Efficient and Versatile Top-k Query Processing for Semistructured Data. VLDB J. 17(1): 81-115, 2008

[26] Fei Wu, Daniel S. Weld: Autonomously Semantifying Wikipedia. CIKM 2007: 41-50

[27] Fei Wu, Daniel S. Weld: Automatically Refining the Wikipedia Infobox Ontology. WWW 2008: 635-644

[28] Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, Wei-Ying Ma: Simultaneous Record Detection and Attribute Labeling in Web Data Extraction. KDD 2006: 494-503