

The GoOLAP Fact Retrieval Framework

Alexander Löser, Sebastian Arnold, and Tillmann Fiehn

Technische Universität Berlin, FG DIMA, Einsteinufer 17, 10587 Berlin, Germany
`firstname.lastname@tu-berlin.de`

Abstract. We discuss the novel problem of supporting analytical business intelligence queries over web-based textual content, e.g., BI-style reports based on 100.000s of documents from an ad-hoc web search result. Neither conventional search engines nor conventional Business Intelligence and ETL tools address this problem, which lies at the intersection of their capabilities. Three recent developments have the potential to become key components of such an ad-hoc analysis platform: significant improvements in cloud computing query languages, advances in self-supervised keyword generation techniques and powerful fact extraction frameworks. We will give an informative and practical look at the underlying research challenges in supporting "Web-Scale Business Analytics" applications that we met when building GoOLAP, a system that already enjoys a broad user base and over 6 million objects and facts.

1 Introduction

Which companies collaborate with Boeing? Are these organizations also collaborating with Airbus or Fokker? Do employees of these companies have a criminal record? Who published the information?

Each day new pages emerge in the Web that may contain a textual representation of facts for answering these questions. Strategic decision makers may frequently research the Web for questions like these. Often answers to these queries might not be published in textual or structured form by Web 'information aggregators' like *Wikipedia.com*, *Freebase.com*, *Trueknowledge.com* or *Yago* [16]. Rather, this rare factual information is hidden in unstructured Web text on a few Web pages of news agencies or blogs. Unfortunately, collecting factual answers from these pages with a general Web search engine is still a dreaded process for a user.

One option to populate a fact base is to crawl a large document collection. For instance *Google Squared* [9] populates its data base with facts from the large corpus of the general Web. The system extracts these facts from tables, lists and from text with open information techniques. However, in [5] we observed that only a small fraction of a large archive *de facto* contains factual information. Hence, strategies that might execute a full scan over the entire archive can drastically waste processing and storage resources (see also [14]). Another option is discovering facts in retrieved pages from ad-hoc keyword search. Unfortunately, this is still a tedious task, since Web search engines do not return aggregated

factual information. The heuristic of a search engine user is: type in keywords as queries, ‘extract’ relevant facts from the top-k documents, filter out relevant facts and compile facts into a structured fact table. Therefore the user typically repeats this process multiple times in order to *complement missing attribute values* and to enhance the chance to *discover unseen facts*.

Significance of our approach. We present GoOLAP, a system that aims to *automate the fact retrieval process from Web search engines*. GoOLAP has three significant aspects: (1) It provides *powerful operators for analytical Web research on textual information*, such as augmenting facts for an object, tracing back the textual origin of a fact or comparing factual information for a list of objects. (2) As a natural alternative to crawling a large proportion of the Web, GoOLAP *interprets user interactions as input to identify missing facts*. These user interactions *trigger a fact retrieval process* with the goal to populate the GoOLAP fact base from Web-scale indices of existing search engines in an ad-hoc fashion. This process is powered by the *FactCrawl* engine that leverages sophisticated *keyword generation techniques* [5, 21] and *page classification techniques* [4] to retrieve only pages that likely contain missing and rare factual information [22]. (3) GoOLAP combines these approaches to drastically avoid crawling, indexing and extracting potentially billions of irrelevant pages. GoOLAP’s human-machine machine generated fact base nearly reaches 6 million facts and objects ; a dimension that is similar to existing community-generated fact bases, such as *DBpedia* [11] or *CrunchBase* [10].

2 Related Work

We discuss relevant related work in the areas of focused fact retrieval from full text indexes, Open Information Extraction and fact search engines.

Keyword query generation with QXtract. The authors of QXtract [2] pioneered work on automatically generating keyword phrases for fact retrieval from full-text indices. The system executes the following stages in a one time learning process: sample seed documents, observe/score phrases in these documents, query a search engine with the most promising phrases and forward the retrieved pages to an information extractor.

Sample seed documents. QXtract requires a small set of manually specified ‘seed’ facts to retrieve an initial set of sample pages for training. The size of this sample influences the precision of the learning process; the authors suggest between 100 and 2500 pages. After sampling, the pages are forwarded to the extractor to identify which documents are relevant for the extraction task.

Observe and score phrases. QXtract utilizes three classification approaches for observing and scoring unigram terms from the documents in the seed sample: An SVM-based classifier, the OKAPI system [25] and the rule-based

classifier Ripper [8] are trained on the set of seed documents. Each approach returns a list of terms, ordered by their significance for classifying relevant and irrelevant documents. The authors of QXtract propose a threshold based technique to assemble and select relevant phrases as keyword queries out of these terms.

Determine document retrieval order. Next, QXtract retrieves a ranked set of documents from the index for each generated keyword query, where the queries are selected from the three lists in a round robin fashion. All documents retrieved for a phrase are forwarded to the fact extractor. The process continues with selecting the next highest ranked phrases from each list and forwarding the resulting documents to the extraction service. It terminates once no more phrases exist in any list. The authors evaluated their approach on the two fact extractors *CompanyHeadquarter* and *DiseaseOutbreak* and achieved a recall of about 20% while processing 10% of the documents in the test corpus. In [14] the authors describe a framework incorporating the idea of automatic keyword generation as an execution strategy.

Phrase generation from SQL queries. Authors of [18] extract and rank potential keywords from attribute names and values of a structured query. They apply a greedy approach that selects terms based on their relevance measures informativeness and representativeness. Our approach makes use of a similar idea: We trust that Web page authors develop a set of common words and grammatical structures to express important factual information in natural language.

Self-supervised keyword generation. In [21] we published an approach that applies a self-supervised keyword generation method to extract meaningful phrases which often correlate with factual information in sentences. Our approach utilizes open information extraction techniques to generalize fact-type-independent lexico-syntactic patterns from sentences [12]. Recently, we presented an extended version of this work on fact retrieval [5], called FactCrawl. Our framework allows for the integration and evaluation of different feature generation methods.

Fact extraction. Multiple projects, such as CIMPLE [26], AVATAR [7], DIAL [13], DoCQS [29], PurpleSox [6] describe an information extraction plan with an abstract, predicate-based rule language. Our approach tries to 're-engineer' common patterns from observing fact extractors and other patterns that appear frequently with factual information on a page as features.

Fact search. Recently, the commercial search engine *Google Squared* [9] populated a fact base with machine extracted information from Web tables and Web lists. That prototype has access to any page in the Google index that contains a table or a list. Contrary, GoOLAP can neither rely on nor compete with this massive amount of data. Rather, we utilize user triggered interactions to generate

keyword queries only for interesting and frequently requested facts. These keyword queries leverage the index of a Web search engine and retrieve only pages that likely contain facts from which we aggressively filter our irrelevant pages. With both techniques, generated keywords and aggressive Web text filters, we leverage the crawling power of a Web search engine, but can drastically limit our fact extraction costs and avoid crawling and indexing potentially billions of irrelevant pages. Google Squared focuses on facts that are either represented as Web lists or as Web tables [9]. Thereby Google Squared may miss important facts in the much more common textual representation. Our extraction approach is complementary since we focus on natural language text only.

Open information extraction and fact search. The Open Information Extraction approach [12] of Google Squared moves the semantic interpretation of fact types to a human user. Contrary, GoOLAP relies on a predefined set of relation extractors. These extractors return well defined semantics for domain specific facts. Thus, this approach allows the aggregation and interpretation of extracted facts by a machine, such as by an OLAP query processor. In addition, these facts can be integrated with user contributed fact bases, such as the project DBpedia [11].

3 Exploring GoOLAP’s Fact Base

This section describes how a user may explore the GoOLAP fact base; see also [20, 19, 5] for a discussion of the research challenges and [23] for an overview of exploratory search:

Interpret. A query typed into a search field can express different intentions. For example, a user has the intention *Augment facts for Boeing*, so he enters “Boeing” into the search field. While he is typing the query, the system returns multiple interpretations in an auto complete dropdown, such as *company Boeing*, *person Boeing* or *product Boeing 747*. The user may select an interpretation for *Boeing* (e.g. *company Boeing*) to send the interpreted query to the system.

Augment. This operation collects and augments information for a particular object. In addition, the system returns images of the selected object and shows links to objects of the same type, such as other companies. Finally, the system returns a list of ranked facts. This process has two steps. (1) Fact type enumeration. The system takes as input the selected object. Each object corresponds to a specific type for which the system looks up known fact types in a system-wide ontology. Currently this ontology holds over 70 different fact types. For instance, for an object of the type *company*, the system returns fact types such as *layoffs*, *competitors* or *products*. (2) Return most interesting facts for each type. For each fact type the system queries the corresponding table for non-empty facts. Next, it ranks facts for each type by the notion of interestingness [17].

Expand. This operation enables a comparative analysis across facts for multiple objects. Consider a user who wants to compare different aircraft vendors, such as *Boeing*, *Fokker* or *Northrup Grumman*. For each object in this user defined selection our system projects a set of facts per type and creates a tabular fact sheet. Objects are displayed as rows, fact types as columns. Each cell lists facts ranked by interestingness. Note that this table represents a non-first-normal-form of the underlying fact base; therefore multiple values per cell may exist.

Trace back. For each fact, the user can access the original document at any point of the assessment process in order to continue his research.

Subscribe. Each user has the possibility to register a profile on the GoOLAP site. This enables a registered user to give feedback to the results by clicking on *agree* or *disagree* buttons or editing factual data in-place. If a user is interested in a particular object, e.g. his own company, he can subscribe to the object by clicking on the star next to the object's name. The system then tries to fetch more data for the subscribed object.

4 User-triggered Fact Retrieval from an Inverted Index

The main goal of our framework is to be able to mine all the facts contained in a document collection while processing as few documents as possible. To solve this task we adapt keyword based document retrieval systems, such as Web search engines, to retrieve only those pages that contain a fact (a factual statement expressed in unstructured text which an information extractor can detect and extract). Our fact retrieval method emulates a search engine users behavior to solve the "inverse retrieval" problem of identifying discriminative keywords for the retrieval of relevant documents. Figure 1 introduces our general fact retrieval process:

Step 1: Interpret and execute user query. The system collects user interactions, determines the intent of the user and executes a query against the local fact base. While a crawl-based fact retrieval execution guarantees that all documents in an archive are processed, an indexed-based execution might miss some relevant documents. For compensating this disadvantage, GoOLAP observes the *completeness* and *rarity* [22] of returned results to identify potentially missing facts. If necessary, GoOLAP schedules a fact retrieval activity (see Table 1). For example, if a user requests facts for an object that is not yet contained in the GoOLAP base, the system will trigger the retrieval of pages that contain facts for the missing object.

Step 2: Generate keyword query to retrieve missing facts. GoOLAP forwards the corresponding textual object representation and the semantic type of the missing fact to FactCrawl [5], a framework that emulates a search engine

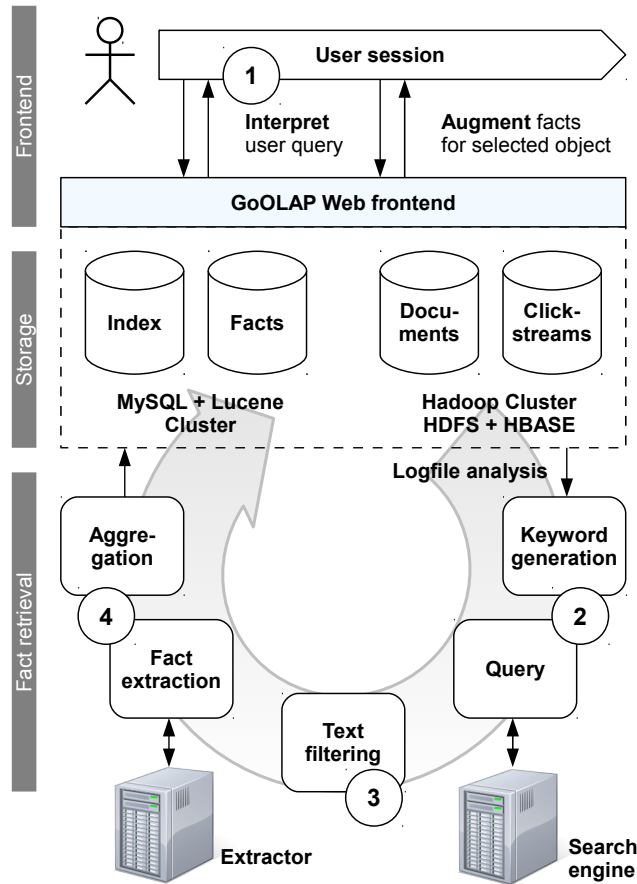


Fig. 1. User triggered fact retrieval from Web search engines. The fact retrieval process is interactively triggered by user interaction, here INTERPRET and AUGMENT.

user’s behavior to solve an inverse retrieval problem. This framework queries a Web search engine with automatically generated keywords, re-ranks the resulting list of URLs according to a novel fact score and forwards only promising documents to a fact extractor. FactCrawl generates keywords using structural, syntactic, lexical and semantic information from sample documents. Thereby FactCrawl estimates the fact score of a document by combining the observations of keywords in the document.

Step 3: Filter out irrelevant pages. GoOLAP utilizes a fact predictor [4]; this technique reliably filters out irrelevant pages and only forwards relevant pages to the GoOLAP fact extractors. Most importantly, our fact predictor is two orders of magnitude faster than the fact extractor. The fact predictor is based on a support vector machine that evaluates pages on a sentence level,

User interaction	Intention: The user...	System activity: The system...	Retrieval activity: FactCrawl...
Click a link that points to GoOLAP	discovers a GoOLAP result page for <i>Company:Boeing</i> in another search engine and clicks on the link	displays the AUGMENT result page for <i>Company:Boeing</i>	retrieves more facts for <i>Company:Boeing</i> , such as <i>headquarter</i> , <i>layoffs</i> etc.
Search with keyword	wants to augment facts for the object <i>Company:Boeing</i> and types <i>Boeing</i> into the search field	interprets the input and displays an autocomplete dropdown with multiple interpretations, such as <i>Company:Boeing</i> , <i>Product:Boeing 747</i> etc.	ranks the order of interpretations when the user selects <i>Company:Boeing</i>
Search missing object	wants to augment facts for an object <i>X</i> that is not yet contained in the GoOLAP fact base	displays a message to inform the user about the empty result and proposes a subscription of <i>X</i>	retrieves facts for the object <i>X</i> with high priority
INTERPRET keyword search via autocomplete	wants to augment facts to the object <i>Company:Boeing</i> displayed in the autocomplete and clicks on it	augments facts to <i>Company:Boeing</i> and displays the AUGMENT page for that object	ranks the popularity of <i>Company:Boeing</i> and retrieves more facts, such as <i>headquarter</i> , <i>layoffs</i> etc.
Show AUGMENT result	requests an AUGMENT result for the object <i>Company:Boeing</i>	displays the top rated facts for <i>Company:Boeing</i> and three arbitrary documents containing information about <i>Boeing</i>	ranks the popularity of <i>Company:Boeing</i> and retrieves more facts
TRACE BACK the document	wants to trace back the textual origin of the fact <i>EADS competes Boeing</i> and clicks on the document symbol	displays a snippet from the source document and explains the origin of a fact by highlighting the sentence where the fact was extracted from	ranks the popularity of the fact <i>EADS competes Boeing</i> and tries to retrieve more evidences of this fact
EXPAND list of objects	wants to compare <i>Competitors of Boeing</i> and clicks on the EXPAND icon in the column header	expands the list of <i>Competitors of Boeing</i> and displays an interactive table view for comparison	retrieves more facts for <i>Company:Boeing</i> and the <i>competitive</i> relation to <i>Airbus</i> , <i>EADS</i> , <i>Fokker</i> etc.

Table 1. GoOLAP collects and interprets more than 30 user interactions that may trigger a fact retrieval process. This table displays user interactions and system activities contained in our demonstration scenario.

where each sentence is transformed into a token representation of shallow text features.

Step 4: Fact extraction. GoOLAP extracts factual information and re-solves objects with home-grown extractors based on [12] and commercial extractors, such as OpenCalais [24]. Overall, GoOLAP provides more than 70 different fact extractors for the domains people, companies, media and entertainment. The system is open to additional fact extractors such as *extractiv.com* or *alchemy.com*. The retrieval process then aggregates the extracted facts and stores them into the local fact base.

Data storage and parallel execution engine. Factual information is small compared to the raw text documents. Therefore we chose to store the local fact base, the ontology, objects and small document snippets of a few 100 bytes in a horizontally partitioned MySQL cluster database on a raid-based 256 GB solid state disc (SSD) array. The technology allows GoOLAP users to retrieve as well as to update and rate facts. In addition, the current user base may generate interactions that trigger the retrieval of several hundred thousand pages daily. We manage the raw textual documents in a *Hadoop* [3] cluster, using HDFS and HBASE. The RDBMS references these documents via unique keys. GoOLAP processes the massive amounts of requests for new facts in a parallel fashion on a cluster of 12 nodes, each running with 4 GB RAM on a 2x 2.8 GHz Intel CPU. The parallel query execution engine bases on Hadoop. We chose to abstract the functional programming interface of Hadoop through the declarative query language *JAQL* [15] and extend JAQL with first-order functions for keyword generation with FactCrawl [5], Web text filtering [4] and fact extraction [24]. JAQL represented objects, pages, facts and interactions with the semi-structured JSON format.

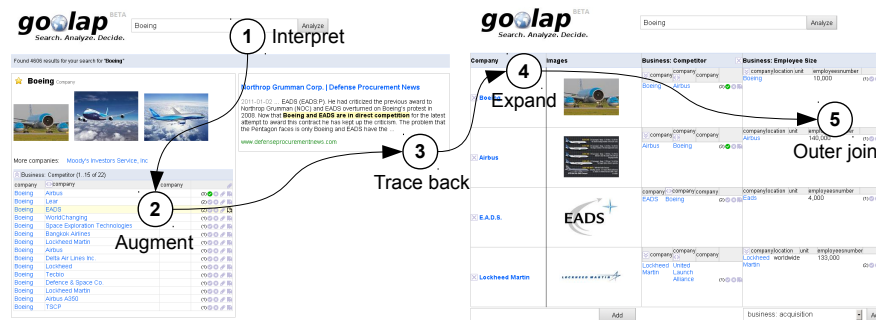


Fig. 2. A typical user session with the goal to collect facts about competitors of company Boeing.

5 Demonstration Scenario

Our interactive online demo shows the interplay of users exploring the GoOLAP fact base and thereby triggering new requests for retrieving missing facts from the index of a Web search engine. Figure 2 shows our demonstration scenario. Our prototype can be reached at www.goolap.info.

Consider an analyst who wants to have an overview over competitors of Boeing. On the GoOLAP start page, she begins to type “Boeing” into the search field (1). The auto complete dropdown returns multiple interpretations of Boeing, picking the company object by default. After sending the query, the following page presents an overview over augmented facts of different types (e.g. business location, products or employment) for the company Boeing and some arbitrary documents that include information about that object. She now opens the table Business: Competitor that shows the top 15 ranked facts about competing companies (2). We suppose she is unsure whether EADS is a competitor of Boeing, thus she clicks on the document symbol on the right of the row to prove evidence for that fact. The next page shows a document from www.defenseprocurementnews.com which explains “Boeing and EADS are in direct competition” (3). She can help ranking the fact by clicking on the agree button or find the sources of other facts in the table the same way. Next, the analyst desires to compare the competitors. She clicks on the button in the column header of the competing companies to expand the list. The view presents a tabular list of competitors of Boeing (4). She can manually remove an incorrect object in the list or add another one that is missing. Suppose the analyst wishes to compare the employee size of the companies. She performs an outer join on the table by adding the column Business: Employee Size from the dropdown menu in order to get a comparable result of facts (5). If there is a wrong result in one cell, she can display alternative values and find a better one. Again, she can trace back the evidence of a fact by clicking the document symbol. She is now happy with her result and saves the table under the name “Aviation Companies”. Now, she can continue her research at

a later time or share the table to other users, allowing them to make their own changes.

6 Research Challenges

GoOLAP presents our attempt in providing a versatile approach for interactive fact retrieval from the Web. We believe that studying the new types of Web usage and the interplay of fact exploration with fact retrieval – a well-known important problem in information retrieval – will open up a lot of new challenges and opportunities:

Ranking fact sources. Only very few domains provide most of the facts in GoOLAP (see also Figure 3). Often these *fact aggregators* aim to collect factual information from news articles or human collaborations. Worse, in [5] we discovered that most fact types appear only in less than 1% of retrieved documents. An

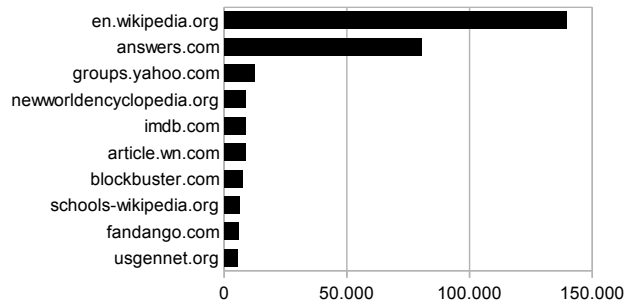


Fig. 3. GoOLAP receives facts from 40,414 different Web domains. This distribution shows the top-10 domains that provided most facts. The X-axis depicts the number of extracted facts per domain.

interesting direction is a systematic integration of these statistical observations into the fact retrieval process; for instance, we could observe domains about fact update cycles or could estimate common replication strategies across domains.

Quality-based fact ranking. GoOLAP ranks different facts for the same object with the measure of interestingness [17]. This measure captures how often authors of retrieved pages mention the same fact for the same object. The design of more meaningful, potentially quality based, ranking metrics desires investigation, such as metrics for ranking ‘rarely appearing’ facts in documents which are often requested through user interactions.

‘Crowd’-based fact sharing and verification. Similar to a data market, users may request the system to collect ‘private’ fact collections and share these

collections with few ‘trusted’ users. How can the system leverage these user interactions and fact collections for discarding incorrect facts? Can we derive a fact quality measure from observing the sharing of fact collections?

Active learning of labeled examples from user interactions. Developing high-quality information extraction (IE) rules, or fact extractors, is an iterative and primarily manual process, extremely time consuming, and error prone. In each iteration, the outputs of the extractor are examined, and the erroneous ones are used to drive the refinement of the extractor in the next iteration. Recently, authors of [1] introduced a provenance-based solution for suggesting a ranked list of refinements to an extractor aimed at increasing its precision. The ability to automatically generate useful refinements depends on the number, variety and accuracy of the labeled examples provided to the system. In most cases the labeled data must be provided by the rule developer. Unfortunately, labeling data is itself a tedious, time-consuming and error prone process. It would be interesting to investigate whether active learning techniques [28] can be combined with techniques for adapting open information extraction to domain specific relations [27] and GoOLAP click streams to present to the developer only those most informative examples, therefore facilitating the labeling process.

7 Conclusion

We introduced GoOLAP.info, a novel class of applications for answering intelligence queries over natural language Web text. GoOLAP provides powerful operators for retrieving, extracting and analyzing factual information from textual representations on the Web. At the core of this service lies FactCrawl, a powerful engine for the retrieval of unseen or missing facts from a Web search engine to a structured fact base. Over time, GoOLAP may evolve to a comprehensive, effective and valuable information source. Unlike systems that utilize a cost intensive crawling strategy (such as Google Squared), GoOLAP elegantly utilizes user interactions, generated keyword queries and text filtering techniques to populate an ad-hoc and incrementally improving fact base. Compared to existing approaches that crawl and maintain very large Web archives, GoOLAP tries to minimize retrieval costs through user triggered fact retrieval.

Finally, we outlined exciting challenges, such as the design of an iterative fact retrieval process, the interpretation of user interaction and automatic keyword query generation. Moreover, GoOLAP gives us access to click streams that allow the academic community to study user interactions and to design novel powerful Web research operators. These challenges should appeal to and benefit from several research communities, most notably, the database, text analytics and distributed system worlds.

References

1. B. L. 0002, L. Chiticariu, V. Chu, H. V. Jagadish, and F. Reiss. Automatic rule refinement for information extraction. *PVLDB*, 3(1):588–597, 2010.
2. E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. In *In Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE)*, pages 113–124, 2003.
3. Apache Hadoop. <http://hadoop.apache.org> (last visited 06/14/11).
4. C. Boden, T. Häfele, and A. Löser. Classification Algorithms for Relation Prediction. In *ICDE Workshops*, 2010.
5. C. Boden, A. Löser, C. Nagel, and S. Pieper. Factcrawl: A fact retrieval framework for full-text indices. In *14th International Workshop on the Web and Databases (WebDB 2011)*, 2011.
6. P. Bohannon, S. Merugu, C. Yu, V. Agarwal, P. DeRose, A. Iyer, A. Jain, V. Kakade, M. Muralidharan, R. Ramakrishnan, and W. Shen. Purple sox extraction management system. *SIGMOD Rec.*, 37:21–27, March 2009.
7. L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan. Systemt: an algebraic approach to declarative information extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10*, pages 128–137, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
8. W. W. Cohen. Fast Effective Rule Induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.
9. D. Crow. Google Squared: Web scale, open domain information extraction and presentation. In *ECIR*, 2010.
10. CrunchBase. <http://www.crunchbase.com> (last visited 06/14/11).
11. DBpedia data set. <http://wiki.dbpedia.org/Datasets#h18-3> (last visited 06/14/11).
12. O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51:68–74, December 2008.
13. R. Feldman, Y. Regev, and M. Gorodetsky. A modular information extraction system. *Intell. Data Anal.*, 12:51–71, January 2008.
14. P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. To search or to crawl?: towards a query optimizer for text-centric tasks. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data, SIGMOD '06*, pages 265–276, New York, NY, USA, 2006. ACM.
15. JAQL. <http://www.almaden.ibm.com/cs/projects/jaql> (last visited 10/31/10).
16. G. Kasneci, F. M. Suchanek, M. Ramanath, and G. Weikum. The YAGO-NAGA Approach to Knowledge Discovery. *SIGMOD Record* 37:4, 2008.
17. T. Lin, O. Etzioni, and J. Fogarty. Identifying interesting assertions from the web. In *18th CIKM Conference*, 2009.
18. J. Liu. Answering structured queries on unstructured data. In *In WebDB*, pages 25–30, 2006.
19. A. Löser. Beyond search: Web-scale business analytics. In *Web Information Systems Engineering - WISE 2009*, volume 5802, pages 5–5. 2009.
20. A. Löser, F. Hüske, and V. Markl. Situational Business Intelligence. In *3rd BIRTE Workshop in Conjunction with VLDB*, 2009.
21. A. Löser, C. Nagel, and S. Pieper. Augmenting Tables by Self-Supervised Web Search. In *4th BIRTE Workshop in Conjunction with VLDB*, 2010.

22. A. Löser, C. Nagel, S. Pieper, and C. Boden. Self-Supervised Web Search for Any-k Complete Tuples. In *EDBT Workshops*, 2010.
23. G. Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49:41–46, Apr. 2006.
24. OpenCalais. <http://www.opencalais.com> (Last visited 06/14/11).
25. S. E. Robertson. On term selection for query expansion. *J. Doc.*, 46:359–364, January 1991.
26. W. Shen, P. DeRose, R. McCann, A. Doan, and R. Ramakrishnan. Toward best-effort information extraction. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1031–1042, New York, NY, USA, 2008. ACM.
27. S. Soderland, B. Roof, B. Qin, S. Xu, Mausam, and O. Etzioni. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102, 2010.
28. C. A. Thompson, M. E. Califf, and R. J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 406–414, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
29. M. Zhou, T. Cheng, and K. C.-C. Chang. Docqs: a prototype system for supporting data-oriented content query. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 1211–1214, New York, NY, USA, 2010. ACM.