DGRC AskCal: Natural Language Question Answering for Energy Time Series

Andrew Philpot, Jose Luis Ambite, Eduard Hovy

Digital Government Research Center
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
{philpot,ambite,hovy}@isi.edu

Abstract

Even quite sophisticated users can experience difficulty navigating large collections of data to locate the answers to their queries. We describe AskCal, a system that employs natural language processing, an ontology, a query planner, and various feedback mechanisms to assist a user in refining his or her query and then in executing and visualizing it. Tracing several interactions with AskCal in the domain of energy time series, we show how a combination of modalities, including ATN parsing of free-form natural language questions, user modification of predefined template queries, and fall-back parsing by picking out landmark terms, support a wide variety of user queries while reducing user query formulation effort. We illustrate the use of feedback mechanisms to guide the user toward regions of the query space where useful data can be found.

1. Introduction

Over that past few years, the Digital Government Research Center[1]'s Energy Data Collection (EDC) project has built a system that provides access to a great deal of data; see (Hovy et al., 2001a; Ambite et al., 2002), or visit http://www.dgrc.org/, click on *Research* and then on *related pages and data*. This data comprises 53,689 time series tables, collected from the Energy Information Administration (EIA), the Census Bureau, the Bureau of Labor Statistics, and the California Energy Commission, and is in a variety of formats, including Oracle and Microsoft Access databases, web pages (HTML), text files, and PDF files.

One of the major research goals of the project is to hide from the user the details of data formats, data locations, and other implementation details, allowing him or her ideally to specify in an implementation-independent way what he or she would like to see, and letting the SIMS access planner (Arens et al. 1996) take care of the actual access planning, SQL query formation, and retrieval operations.

However, we encountered the following very common problem: the more the system hides complexity by making assumptions and operating automatically, the less flexibility the user can exercise. This limits the system's utility unnecessarily. An expert user, or even a non-expert user with a clear idea of what he or she needs, must be able to specify requirements quite exactly. So when the user does not know the particulars of the domain, the system has to support on-the-spot browsing and learning, in order to allow the user to specify queries as precisely as desired, and let the system handle the rest. Since the EDC data is characterized by 10 dimensions (gasoline type, area, period and frequency of measurement, unit of measurement, agency, etc.), only the most expert user is likely to know all the allowable values for each aspect of the data.

In such a large system, with complex sets of data, providing easy to use support is not simple. In general, the system builder faces a dilemma when it comes to helping the user build queries:

---

[1] The Digital Government Research Center (DGRC; www.dgrc.org) was established in 1999. The DGRC consists of faculty, staff, and students at the Information Science Institute (ISI) of the University of Southern California and Columbia University's Computer Science Department and its Center for Research on Information Access. The mandate of the DGRC is to conduct and support research in key areas of information systems, to develop standards/interfaces and infrastructure, build pilot systems, and collaborate closely with Government service/information providers and users.

- One may provide a highly expressive query access language, such as SQL. Unfortunately, one thereby loses non-expert users. In addition, when the data space is non-homogeneous (i.e., when data are clustered at certain sets of parameter values), it is difficult to guide expert users toward areas where data will be found.
- Alternatively, one may provide a less detailed, more intuitive query construction paradigm, either graphical or natural language based. Unfortunately, although one can in this case help guide the user through the data space, one loses expressive power in various ways, including underspecificity, scoping (does an "all " apply to just one clause or many? Which?), word ambiguities, and the inability to express complex interdependencies.

In order to learn which approach would provide optimal user support for EDC, we first incorporated the domain model browsing interface DINO (http://edc.isi.edu:8000/dino) with which the user can explore the data space. We then implemented three different modes of query formation for the system:

1. **raw SQL** creation using a text editor,
2. a **cascaded menu** interface in which the user can select appropriate parameter values,
3. a semi-free form **natural language (NL) interpreter** to which the user can state queries.

We learned from this work that no single modality is optimal. In most cases, query construction is a process of interactive 'deepening' toward more specificity, and at different times in the process different approaches are appropriate. In this paper, we describe our resulting hybrid modality, namely **initial natural language questions followed eventually by a structured menu-based query interface**. This dual approach, implemented as AskCal, has the following characteristics.

Initially, by simply typing an English question, the user can cover a lot of ground in aspects he or she is more sure about, without having to worry about query language terms, formulation, and syntax. For example, a user may initially ask "How much is gasoline in California?" The system is able to interpret that the user wants the *price* of gasoline. However, the question is very underspecified with respect to the data that the system has access to. What type of gasoline does the user want: premium, regular, reformulated, oxygenated, …? What type of price: consumer price, wholesale price, …? At what time point should the gasoline price be given? These choices and distinctions have to be shown to the user to help him or her understand the domain and specify a query that can be answered by the system.

Subsequently, when the user needs more guidance, AskCal reformulates the English question into a standard form and displays it together with the remaining unspecified parameter fields and their possible values, all still phrased as an English question. Now the user is free to continue typing English or to point and click menu items. Such structured guidance helps both the user (to know what is allowed) and the system (to handle underspecificity and ambiguities).

An important point of feedback is the richness of the query as currently specified. At any point, the system displays the number of data tables specified by the query. Usually, when the number is too high (say, 10,000), the user chooses to continue refining the query by selecting more precise values or ranges. When the number is too low (say, zero), the user can see he or she has landed into an area where no data has been collected, and can 'back out' in various directions by relaxing parameters. This simple feedback allows the user to explore the data space quite effectively.

Still, when the data space is large, it may be difficult for a non-expert user to locate regions in the data space where data does exist. We therefore implemented a set of standard but typical questions, also displayed as a menu in AskCal, and listed in Section 3.2.

Figure 1 shows a natural-language query input by the user and the system's reinterpretation and menu-based additions. Figure 2 shows the interface after the user has explored the terms proposed by the system and selected some values. The system shows the number of series covered by the user query. When the number focuses on a small number of time series, the system also provides the values of the dimensions that differentiate one series from another (the product dimension in the example in Figure 2).

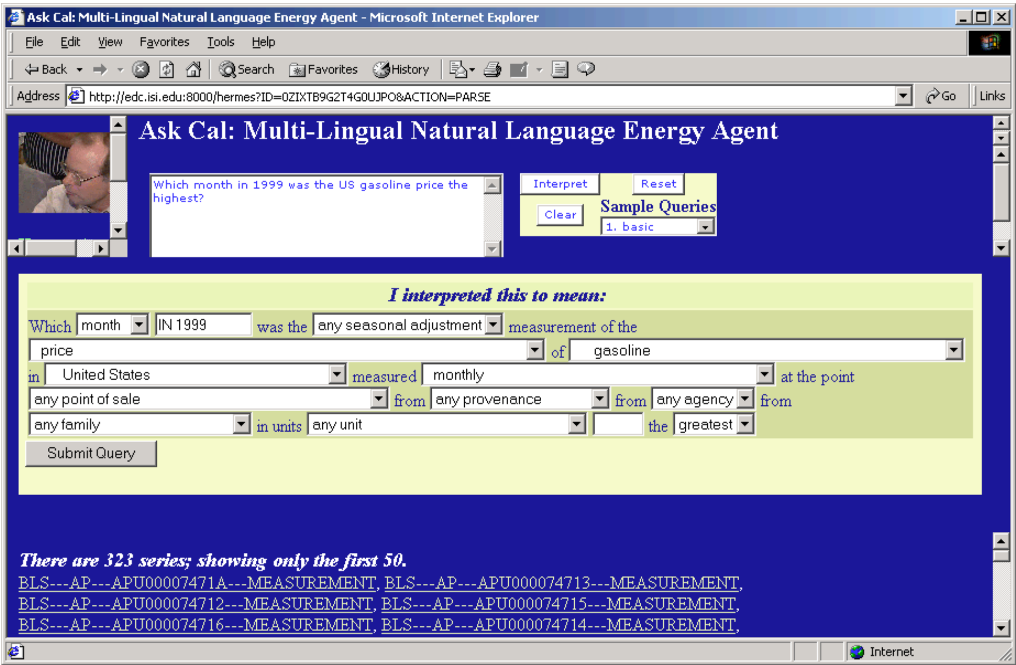

Figure 1. AskCal interface, showing the initial user query (top pane), the system interpretation (middle pane), and the number of time series covered by this query.

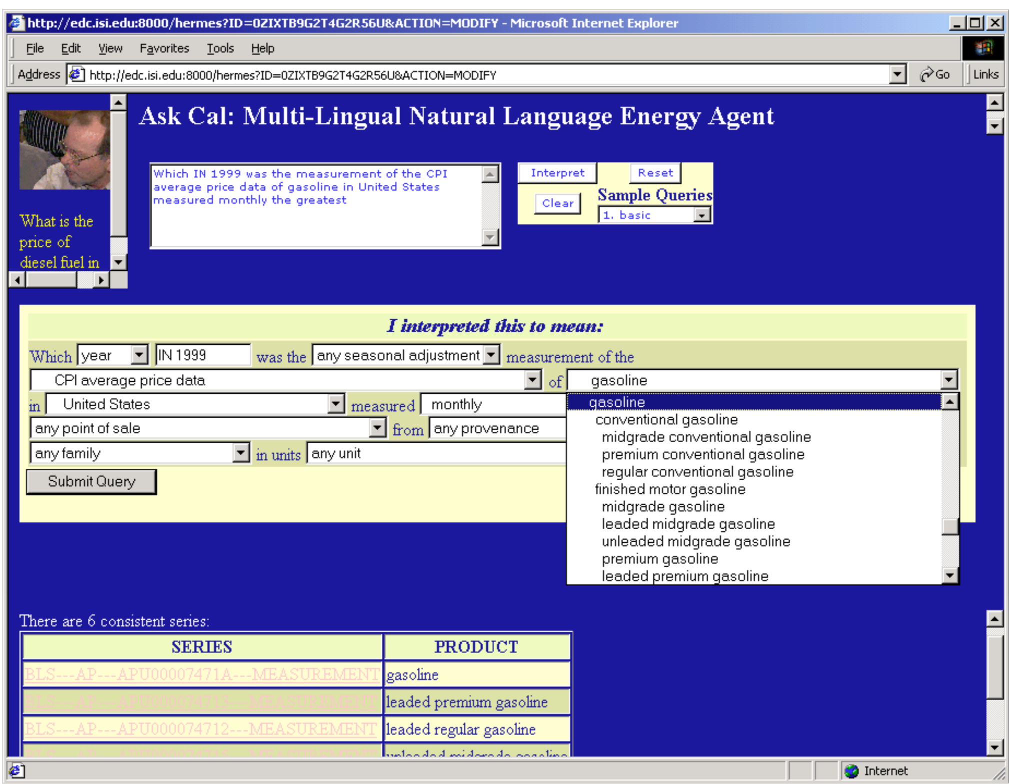

Figure 2. User focuses on time series of interest by making additional choices, which are reflected by the NL text (top pane). The remaining time series display the differentiating dimension: in this case, 6 time series differentiated by the type of gasoline product.

2. Related Work on Querying and Question Answering

The fundamental way to extract data from a database system is using a query language. In the case of relational databases, SQL is the standard query language. However, SQL requires the user to learn a non-trivial syntax and semantics. Naïve users often have difficulty writing syntactically correct queries as well as expressing the semantics of the query using SQL constructs. Thus, database and usability researchers have investigated how to provide more intuitive and convenient methods for database querying. The two major approaches have been visual query languages and natural language processing.

Visual query languages provide a graphical interface to describe a query. The user constructs the query by selecting different elements by point-and-click as opposed to writing an SQL string. This approach is more time-efficient and less error-prone than writing SQL directly; experiments suggest that the user is often able to express his or her queries more accurately. One of the first systems to facilitate query construction is Query-by-example (Zloof 1997). We refer the reader to (Catarci et al. 1997) for a survey and to (Catarzi 2000) for some recent work. However, since the user still has to understand the semantics of the graphical primitives used to construct queries, this approach still presents a barrier for naïve or occasional users.

Natural Language access to databases has been a topic of great interest since the early 1970s; for an overview see (Androutsopoulos and Ritchie 1995). Some early well-known systems were LUNAR (Woods et al. 1972), LADDER (Hendrix et al. 1978), and CHAT-80 (Warren and Pereira 1982). But despite considerable research interest in the 1980s, the kinds of problems outlined in Section 1 remain unsolved, and commercial NL interfaces to databases are not common, although see BBN's PARLANCE (BBN 1989), Natural Language Inc.'s NATURAL LANGUAGE (Manferdelli 1989), and IBM's LANGUAGEACCESS

Over the past few years there has been a resurgence of interest in question answering in the Information Retrieval (IR) and Natural Language Processing (NLP) research communities. The task is generally defined as finding short (50-byte) answers (factoids) to questions, by reference to NIST's TREC collection of 1 million newspaper texts. Periodic evaluation competitions show the best performance to be around 50% (Mean Reciprocal Rank score). At ISI, we have been building the Webclopedia, a QA system that combines IR and NLP methods (Hovy et al. 2001b; 2002). The Webclopedia Question/Answer Typology at http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html contains 140+ types of question/answer such as *WhyFamous, Y-N-question, BirthYear*. It is one of the most extensive QA typologies available.

3. AskCal

Designing a natural language interface to complex data systems such as EDC involves several major decisions. The first is what types of queries the system is going to support. The expressiveness of the queries impacts both the kind of natural language that the system must accept as well as how the natural language is mapped to the formal query language supported by the data system. Some limitations in both respects are needed since unrestricted natural language understanding is an open problem; even if the natural language is correctly interpreted it can generate queries that are too powerful for a database language to execute efficiently. Inspired by the Webclopedia Question Answering Typology, our approach is to focus initially on a limited set of query types and grow the coverage of natural and query languages incrementally. Our natural language interpretation is easily extensible to parse new queries. The second decision is what kind of natural language understanding technology to use. Since we expect our system to run on a busy web server, we cannot afford to produce sophisticated parses of the queries. Our implementation uses augmented transition networks (ATNs) that provide good coverage of typical questions, are easily extensible, and execute very efficiently. Finally, the natural language understanding is just one part of the interaction with the user. The AskCal system allows the user to explore and refine the query using NLP, menu-based choices, and data coverage hints.

In the remainder of this section, we first discuss the query types that AskCal currently supports. Then we describe our ATN-based natural language query interpretation. Finally we provide an overview of the AskCal interaction algorithm.

3.1 Query Types

In order to understand which types of queries to implement first, we conducted a survey among potential users. This resulted in a set of 14 query types that serve both to aid question formation and to guide users toward available data. The following is a list of representatives of each query class:

- **Basic1**: "Please give me the premium conventional gasoline price in Texas measured monthly."
- **Basic2**: "How much is the gasoline in Hawaii?"
- **When**: "When was the premium gasoline price in Texas measured monthly higher than 155 cents per gallon?"
- **Which**: "Where is the most expensive gasoline found? "
- **Counttimes**: "How many months did California pay more than $1.25 for oxygenated gasoline since 1990?"
- **Listtypes**: "How many types of premium gasoline are there?"
- **Describe1**: "What is the octane rating of midgrade unleaded gasoline?"
- **Describe2**: "What is nuclear electricity?"
- **Extremum**: "Which month in 1999 was the US gasoline price the highest?"
- **Howmuch**: "How much premium gasoline was sold in the US in 1999?"
- **Selecttotal**: "What is the total dollar value of gasoline refined in Louisiana last year?"
- **Lasttime**: "What was the last week since 1995 when regular unleaded gasoline was less than $1.10 per gallon in California?"

All these types are re-formulated into a canonical, synthetic, paraphrase format, one that is unlikely ever to be created directly by a user. This format also needs to be parsable, however, since the user could ask for its re-interpretation in the NL Query pane (see Figures 1 and 2). So this synthetic paraphrase constitutes a 15th type:

- **Synthetic**: "Please give me the date and value for the seasonally adjusted measurement of the price of unleaded gasoline in California measured weekly at the point production from WWW from BLS from Urban Consumers CPI in units dollars per gallon on April 11, 1968"

3.2 ATN parsing

We have implemented an ATN parser, adapted from the one in (Graham 1994). The ATN grammar comprises 341 nodes, and the domain model covers 55000 concepts, which are represented using our 400 dimensions, taken 10 at a time. A small fragment of the ATN specification is shown in Figure 3. Each doubled oval calls a subnetwork, and each subnetwork recognizes one Noun Phrase (NP) or Prepositional Phrase (PP) by matching at each arc the required words (e.g., "which", "in, from, into…"), parts of speech (e.g., Prep, Det), or nothing (empty transition 0). At each point a matched arc contributes appropriate content to the output structure (not shown for reasons of complexity). On exiting a subnetwork or network, the accumulated structure is inserted into the calling network's structure.

The matcher (called for each arc) returns both the original token and the match to the transition rule. Consider case folding: sometimes one has to preserve the case supplied (say, for a person's name) and sometimes one needs to use the matched token (say, for a state name). This mechanism allows the transition to use whichever it needs.

We adapted Graham's design in several ways. Principally, we dropped Graham's implementation of non-determinism using a macro layer of continuation passing. Instead, we use backtracking depth-first search, which we found to be more efficient both at run-time and at compile time. Other extensions to provide better expressivity include support for multi-word tokens, support for optionally matching a category modulo stemming, wild card matching, and matching via applying an arbitrary function to the token. This latter is useful for recognizing tokens via a programmatic filter: e.g., numbers 1900–2100 can be parsed as year expressions.

AskCal also includes a **fallback** procedure. In this mode, most of the high-level categories are parsed 'in parallel', independently of each other: this allows random-looking material before and after an expression of interest to be accepted. High-level categories include all the 10 dimensions (area, agency, product, provenance, point of sale, frequency, seasonal adjustment, quality, family, unit) plus date/interval expressions, comparison expressions (greater/less), and extremum expressions (the largest/cheapest/maximum). The fallback mode could be considerably more sophisticated, and hence can be easily fooled. A typically comic error is to confuse state abbreviations with prepositions/conjunctions (IN, OR). Obvious heuristics exist, but the free-form input prevents being too smart.

The ATN parser fills the values of the query template associated with each query type. The instantiated query template is sent to the data system for execution.

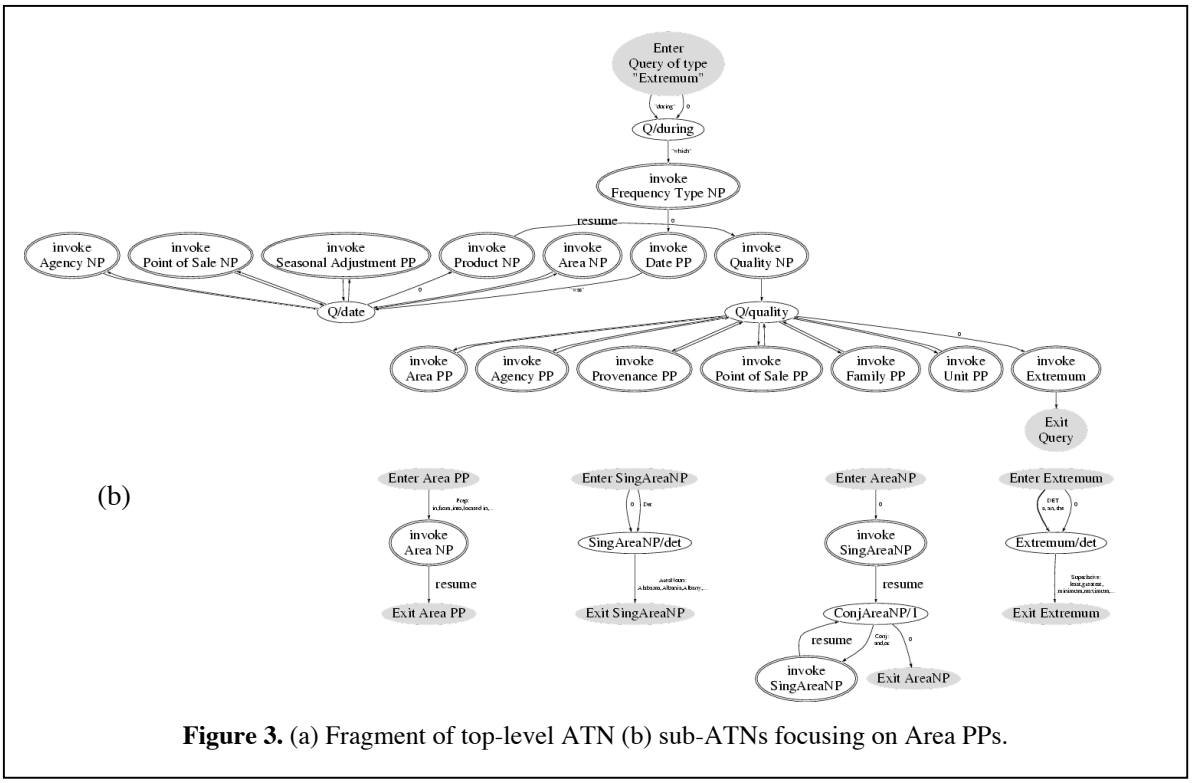

Figure 3. (a) Fragment of top-level ATN (b) sub-ATNs focusing on Area PPs.

3.3 AskCal Algorithm

AskCal implements the following algorithm to structure the interaction with the user (see Figures 1 and 2):

1. The user specifies a query in natural language, possibly by modifying one of the sample queries in the menu.

2. The ATN parser produces all possible parses of the query, filling the format of one of the 15 query targets or the catchall (synthetic) format. If no query target can be found, fallback parsing of all fragments is performed.

3. The parser performs a heuristic evaluation to choose the parse, using the following rules:

- Prefer one of the 14 query targets over the synthetic target (which is preferred over a fallback parse).
- Some query type ATNs include epsilon (empty) transitions that introduce defaulted or implicit information, e.g., assuming certain words are missing. Parses that default fewer items are preferred.
- Sometimes a syntactically legal parse might attempt to set a category more than once. For example, the sentence "Show me gasoline in Texas" has the silly parse "Show ME (= Maine) gasoline in Texas", which has two bindings for the AREA category. Parses with fewer such duplicates are preferred.
- If multiple parses score the same, one of the top scorers is selected arbitrarily. If nothing is recognized, the default query is effectively "Show all gasoline measurements".

4. The successful parse is paraphrased back into English and displayed (see top pane of Figure 2). The paraphrase is generally not identical to the supplied natural language query: we avoid implicit references/defaults: we prefer using phrases with introduction words ("from Texas", "measured annually") over prefix modifiers ("Texas", "annual"), and we choose a standard order for clauses.

5. Simultaneously, AskCal analyzes the instantiated query template to determine the applicable answer type. Currently it can provide two classes of answer: concept definitions or descriptions (Listtypes, Describe1, and Describe2 above) and data series (the others). Concept definitions and descriptions are answered by retrieving textual information directly from the appropriate concepts in the Domain Model. Retrieving data series requires the description logic reasoner of the EDC system (Ambite et al. 2001). The system constructs a concept using the

dimensions specified in the query template (area, product, etc.). The reasoner classifies the concept expression into the Domain Model and returns the set of most specific time series consistent with such input concept. For example, a query for price of gasoline in Texas will also return wholesale price for unleaded gasoline in Texas. The resulting set of time series (is presented to the user (see bottom panes of Figures 1 and 2). If the query covers many time series, only the number of relevant series is shown. If the set is small, each series appears in a table showing the dimensions that differentiate one series from the others (see bottom pane of Figure 2).

6. The user may modify the original NL sentence, by editing the NL paraphrase, or by selecting values from the menu elements of the interpreted query (see middle panes of Figures 1 and 2) and repeating the process from step 2.

7. The user finally selects one of the identified series. The system focuses on that series, showing all the dimension values that define the series. The user can retrieve the time series as a table or graph.

4. Conclusion

We have described AskCal, a natural language interface to data integrated from a variety heterogeneous sources about energy products distributed among several government agencies. AskCal is one of the interfaces of the Enegy Data Collection Project that includes direct SQL, menu-based, and ontology-browsing interfaces providing complementary interaction modes. Each interaction mode addresses the needs of a different user group. The NL interface is most indicated for the naïve or the occasional user and provides a simple mode of interaction and guidance for the user. In spite of limitations in the range of queries accepted, our experience indicates that AskCal is valuable for web portal applications such as the integrated statistical data system of the DGRC's EDC project.

References

Ambite, J.L., Y. Arens, W. Bourne, P.T. Davis, E.H. Hovy, J.L. Klavans, A. Philpot, S. Popper, K. Ross, P. Sommer. 2002. A Portal for Access to Complex Distributed Information about Energy. *Proceedings of the NSF's National Conference on Digital Government dg.o 2002*. Los Angeles, CA.

Androutsopoulos, I. and G. Ritchie. 1995. Natural Language Queries to Databases—An Introduction. *Natural Language Engineering* 1(1):29–81.

Arens, Y., C.A. Knoblock and C.-N. Hsu. 1996. Query Processing in the SIMS Information Mediator. In A. Tate (ed), *Advanced Planning Technology*. Menlo Park: AAAI Press.

BBN Technologies. 1989. BBN Parlance Interface Software—System Overview.

Catarci, T, M.F. Costabile, S. Levialdi, C. Batini. 1997. Visual Query Systems: Analysis and Comparison. *Journal of Visual Languages and Computing* 8(2):215–260.

Catarci, T. 2000. What Happened when Database Researchers Met Usability. *Information Systems* 25(3):177–212.

Graham, P. 1994. *On Lisp*. Prentice Hall.

Hendrix, G., E. Sacerdoti, G. Sagalowicz, and J. Slocum. 1978. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2):105–147.

Hovy, E.H., A. Philpot, J.L. Ambite, Y. Arens, J. Klavans, W. Bourne, and D. Saroz. 2001a. Data Acquisition and Integration in the DGRC's Energy Data Collection Project. *Proceedings of the NSF's National Conference on Digital Government dg.o 2001*. Los Angeles.

Hovy, E.H., U. Hermjakob, and C.-Y. Lin. 2001b. The Use of External Knowledge in Factoid QA. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD, pp. 166–174.

Hovy, E.H., U. Hermjakob, and D. Ravichandran. 2002. A Question/Answer Typology with Surface Text Patterns. *Proceedings of the Human Language Technology Conference (HLT)*. San Diego, CA.

Manferdelli, J.L. 1989. Natural Languages. *Sun Technology*, pp. 122–129.

Warren, D. and F. Pereira. 1982. An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *Computational Linguistics* 8(3–4):110–122.

Woods, W.A., R.N. Kaplan, and B.N. Webber. 1972. The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, MA.

Zloof, M, 1977. Query-by-example: A Data Base Language. *IBM Systems Journal* 16(4):324–343.