Omnibase: A Universal Data Source Interface

Boris Katz, Deniz Yuret & Sue Felshin

Artificial Intelligence Laboratory
Massachusetts Institue Of Technology
Cambridge, Massachusetts 02139

http://www.ai.mit.edu

The Problem: The wealth of information available today by electronic means does not neatly fall into a single format. Not only are there disparate formal database languages, but the largest source of electronic information, the World Wide Web, has no formal data structure at all. Traditional keyword-based Web search engines have "solved" this problem by treating all data as raw text and performing literal character matching only; these searches generally miss much data they should find and find much data they should miss. Other information retrieval (IR) programs have addressed the problem by accessing only a single database or Web site. To access *all* the available information, an IR engine must be able to access the data on the Web and/or in other databases despite their differing formats.

We believe that natural language is the most natural form of communication and information access for humans. Our START Natural Language System [1] is a natural-language-based information retrieval system. One reason why full-text natural language parsing is not yet possible is the difficulty of recognizing certain constructions, such as proper names, which are often intractable from the standpoint of syntactic analysis. While a natural language system can implement specialized new rules to handle proper nouns, these rules typically are linguistically vague (N => name*, e.g., "John Paul") or ambiguous (N => NP, N => VP, N => [anything], etc., e.g., "Rolling Stones," "Gone with the Wind," "Nicely Nicely"), slowing processing drastically and making it difficult to avoid analyzing everything as a potential name. If the natural language system were to store these tokens, its lexicon would become cluttered and unwieldy. Moreover proper nouns are constantly being coined or dropping from use, making it difficult to keep a lexicon up to date.

Motivation: While it is possible to handle the problem of disparate database formats by integrating each format separately into an IR engine, it is far more efficient to design a modular, uniform database which serves as a front end to all other databases. We have implemented Omnibase, a program which provides a uniform interface to databases which store information about objects. Because databases can vary so widely in their capabilities and functions, it is impossible for Omnibase to support all capabilities for all databases. However, essentially every database supports the simple abilities to store and retrieve entities with attributes, so Omnibase provides a uniform interface for this retrieval.

Since Omnibase must store entities anyway, to retrieve their attributes, it makes sense to use Omnibase additionally to detect entities in text. Omnibase serves as a "symbol table" (sometimes called a "gazetteer") of entities. By serving as a lexical adjunct to START, Omnibase allows START to efficiently parse queries containing entities such as proper names regardless of their form.

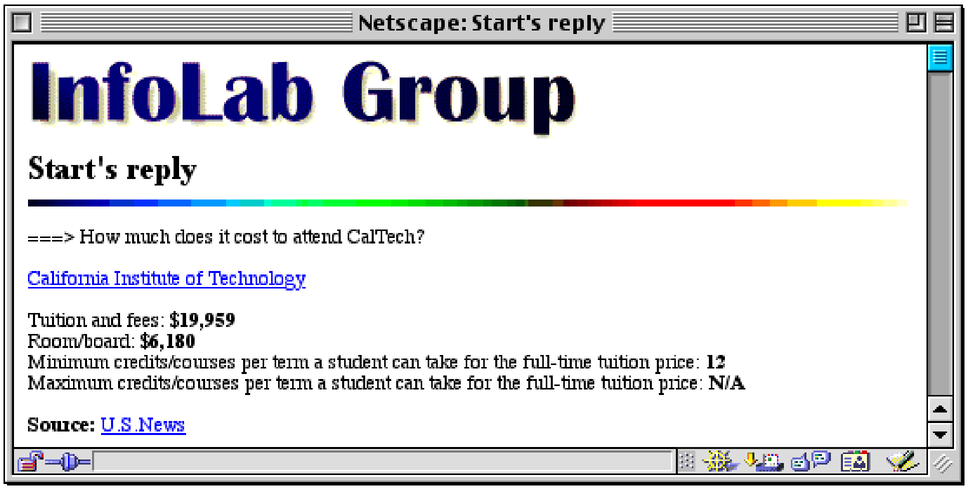
Previous Work: See [2, 3]. Omnibase has taken over Blitz's symbol-table-based capabilities and calls Blitz to make use of its remaining heuristic capabilities [4].

Approach: Omnibase uses two main procedures for storing data: 1) It can add entities to its internal database. Entities are identified by their **class** and **symbol** name. 2) It can add scripts to its internal database for retrieving attributes of entities. Scripts are associated with classes.

Omnibase has two main procedures for retrieving data: 1) Given a query string, it can detect entities within the string, allowing mismatches in case and punctuation, and recognizing synonyms. 2) Given an entity designator and an attribute name, it can retrieve the entity's value for that attribute by running the class script associated with the attribute.

Classes are specific to data sources, and therefore attribute scripts are customized to particular data sources. If one data source is a search URL, attribute scripts for classes in that data source will construct appropriate URLs, post them to the Web, and return the result, possibly parsing the result to return only part of it. If another data source is one or more Web pages with data for all entries expressed in similarly formatted HTML, attribute scripts for classes in that data source will retrieve the Web page and parse the HTML to find the correct segment of the Web page.

The core of Omnibase is deliberately minimal in size. It is implemented in Guile [5] and stores local data in an SQL database. Guile has all the advantages of Lisp and C combined, along with regular expression capabilities. Attribute scripts are extraordinary easy to write; in fact they are so easy to write that we have abandoned our LaMeTH tool for generating HTML-parsing scripts [6].



Impact: We have integrated Omnibase into the START Natural Language System [1]. Omnibase allows us to quickly and conveniently augment START's knowledge base with Web and other data sources. It is no longer necessary to compromise START's modularly with large amounts of database-specific code. Students can learn in hours how to write Omnibase scripts, substantially reducing the time it takes to integrate a new data source. Omnibase has significantly increased the quantity and diversity of data which START can access and queries which it can answer. Most recently, Omnibase played a significant role in START's participation in DARPA's High Performance Knowledge Bases program, enabling the rapid construction of START knowledge bases concerning military capabilities and geographical information about Middle East countries for the purposes of question-answering.

Future Work: Omnibase is currently designed to handle only one type of database retrieval—retrieval of an attribute of an entity. For exploratory purposes, we have added the ability to pass raw SQL queries to Omnibase, which passes them through to its underlying SQL database. We would like to codify these more-complex retrievals into a formal language, to be implemented as a second layer which will act as an intermediary between outside programs and Omnibase. In other words, Omnibase should act as an intermediary translating formal syntax into data-source-specific syntax, while "Language Two" should act as an intermediary translating formal semantics into data-source-specific semantics.

Research Support: This research is funded by DARPA under contract number F30602-00-1-0545 and administered by the Air Force Research Laboratory.

References:

[1] B. Katz, et al. Information Access Using Natural Language. *MIT Artificial Intelligence Laboratory Research Abstracts, September 2000* (this volume).

[2] B. Katz, et al. Blitz: A Preprocessor for Detecting Context-Independent Linguistic Structures, Parts 1 and 2, Heuristics and Symbol Tables. *MIT Artificial Intelligence Laboratory Research Abstracts, September 1998.*

[3] B. Katz, D. Yuret, J. Lin, S. Felshin, R. Schulman, A. Ilik, A. Ibrahim, and P. Osafo-Kwaako. Integrating Web Resources and Lexicons into a Natural Language Query System. *Proceedings of IEEE Multimedia Systems*, Florence, Italy, 1999.

[4] B. Katz, et al. Blitz: A Preprocessor for Heuristically Detecting Context-Independent Linguistic Structures. *MIT Artificial Intelligence Laboratory Research Abstracts, September 2000* (this volume).

[5] Free Software Foundation. Guile: Duct Tape for Bits. http://www.gnu.org/software/guile/index.html.

[6] B. Katz, et al. LaMeTH: An Integrated System for Content-based HTML Extraction. *MIT Artificial Intelligence Laboratory Research Abstracts, September 1998*.