

Using Open Data Collection for Intelligent Software

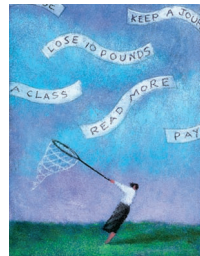
David G. Stork, Ricoh Silicon Valley

General purpose software for speech recognition, natural language understanding, handwriting recognition, common sense reasoning, computer vision, scene analysis, and so on—“intelligent software”—has been a dream for decades. After a period of zealous hype during the early days of artificial intelligence, researchers now appreciate the profound difficulties involved in developing such software. Nevertheless, the effort to develop intelligent software led to many advances in software technologies, such as languages like Lisp and Prolog, expert systems and functional programming methodologies, and fundamental algorithms in pattern recognition, machine learning, search, and automated reasoning.

One important lesson garnered from the decades of work on this concept is that we need large data sets to build intelligent software. This lesson, and several software trends related to the open source movement and Internet computing, suggests that we are poised for another step in the evolution of software technologies. While it is too early to speak with certainty, increasingly compelling evidence indicates that new software methodologies that draw on this research will address the problem of collecting and using data openly contributed over the Internet.

THE NEED FOR TRAINING DATA

For many problems in intelligent systems, building the core software generally requires making an algorithmic



Open data collection offers immense opportunities for generating the large data sets necessary to develop and train intelligent software.

model and training that model with data. For example, virtually all speech recognition systems rely on some form of Hidden Markov Model (HMM) as the model of a spoken utterance, such as a word. An HMM has a number of free parameters or variables, and for accurate speech recognition these parameters must be trained using large amounts of data extracted from actual spoken words. Clearly, accurate recognition requires large training data sets—the larger the better.

In both the academic and commercial realms, the best systems for optical character, speech, face recognition, and so on, are those trained with the largest data sets. Leading optical character recognition companies devote significant resources to collecting and labeling examples of hand-

written and printed characters. Likewise, the Linguistic Data Consortium at the University of Pennsylvania has dozens of employees collecting and labeling linguistic data, which is then made available to developers of speech and language systems. These and innumerable other projects show that collecting very large, high-quality data sets is vital to intelligent software development.

In many cases, the required data is informal—that is, known by everyone who can read, speak, hear, or has common sense. Such informal data differs significantly from formal information contributed by expert programmers through traditional open source projects, such as code for device drivers, schedulers, and file systems in the Linux operating system.

SOFTWARE AND ALGORITHM TRENDS

Beyond this need for data, several trends point to a new approach to creating intelligent software:

- Massive expansion of the Internet, particularly the growing number of nonspecialist users, sometimes referred to as “netizens,” on the World Wide Web. The Web now provides a low-cost framework for collecting data contributed by large numbers of users.
- The success and growing acceptance of open source development methods and the software that results from them, such as the Linux operating system, Emacs text editor, Mozilla Web browser, and Apache Web server.
- Improvements in distributed software and tools that support collaboration over the Internet—both among experts and, increasingly, among nonexperts.

- The rise of machine-independent languages for Web computing, particularly Java.
- Refinement of techniques in pattern recognition, machine learning, grammatical inference, data mining, and other disciplines central to the development of intelligent software. A growing number of experts believe these methods are now sufficient for developing useful intelligent software in many problem domains. We lack, however, adequately large high-quality data sets. Even debating experts agree that we need large data sets to distinguish and choose among competing models.
- Relentless reduction in the cost of computers and digital storage, as anticipated by Moore's law that computing capacity doubles every 18 months.

These trends reveal a shift toward greater open networked collaboration.

OPEN DATA COLLECTION

A new approach to collecting the data required for building components of intelligent systems must leverage these trends to extend traditional open source methods. Such an approach would let nonexperts contribute informal data over the Internet. One application of this concept would be software to recognize isolated handwritten characters, as might be used for reading hand-printed forms or zip codes on some postal envelopes. First, a large collection of unlabelled handwritten characters is scanned from representative documents and placed in a database on a host site on the Web. Next, contributors use standard Web browsers to visit this site, then identify or label each character by clicking on one of 10 corresponding buttons (0, 1, 2, and so on). Such responses are then aggregated and used as training data to improve the classifier software resident at the host. The resulting classifier software and data can be downloaded by anyone over the Web, through an open source license.

In a related approach, contributors could download an open source classifier and use it on their local desktop machines. Recall that virtually all classi-

fier software, such as speech recognition programs, is continually trained and refined by users. Such training data would be stored locally on contributors' desktop machines and periodically submitted automatically over the Web to the host site. At the host site, the training information that all participating contributors need is used to train the parameters in the

While traditional distributed projects exploit the unused power of many networked computers, open data collection instead harvests the information from contributors' minds.

classifier. These improved parameters are then automatically distributed to each contributor, thereby giving superior classifier accuracy. In this way, the full system exploits the efforts of many participants—the more participants, the faster the recognizer improves.

OPEN DATA COLLECTION VERSUS TRADITIONAL METHODS

This form of open data acquisition differs from traditional methods of open source, data mining, distributed computing, and collaborative decision making in several ways. First, while traditional open source projects collect formal knowledge from experts, which then appears in released software, open data collection yields informal data from many nonexperts, which then appears only indirectly in the released software after training. Final decisions in traditional open source projects are arbitrated by an expert; in open data collection, the infrastructure software accepts or rejects much information automatically.

Open data collection also differs from data mining. Data mining seeks to uncover information from an existing data set of essentially fixed size. In open data collection, however, contributors willingly give new data, and the data set grows as long as necessary. Moreover, for many applications there simply are no

appropriate data collections that could be mined.

Fundamental differences exist between open data collection and distributed computing projects such as SETI@home, prime factorization, and decryption attacks. While traditional distributed projects exploit the unused power of many networked computers, open data collection instead harvests the information from contributors' minds. Collaborative voting, auctions, and decision systems such as the Iowa electronic markets indeed pool information from many users, but such voting yields a single decision or opinion rather than intelligent software that can be used in a wide range of novel problems.

Although several Web projects bear some similarities to open data collection, all differ in at least one fundamental way. One similar project deserves mention, however: The Newhoo! open Web directory project has contributors propose keyword and index information about Web pages, which is then reviewed by volunteer referee-editors and used for improved Web searches. However, because Newhoo! does not exploit machine learning, it differs from the current open data acquisition and training scheme.

NEW SOFTWARE METHODOLOGIES

The various software methodologies that will utilize open data collection must build upon those techniques refined by the traditional open source community to support collaboration. They will also rely on techniques from Web computing and will likely use machine-independent languages such as Java.

The new software methodologies should be:

- *Vigilant.* The data collection software running on the server should automatically detect the submission of poor-quality data by unreliable contributors or hostile attackers. Acting much as a sentry, the software will discourage submissions from hostile attackers and reward friendly contributors, encouraging them to continue providing data. This behavior will require security models based on the data submitted

itself, the state of the classifier software, information concerning past submissions by the contributor in question, and so on.

- *Interactive.* The software should respond rapidly to submissions, particularly with regard to rejecting bad data. The constraints from interactivity are particularly strong if a game interface is used. Imagine compelling games in which each user click contributes information to the growing data set. The software should support local caching of patterns, responses, and code-driving interfaces on each contributor's machine.
- *Distributed and scalable.* The software should support large numbers of contributors and periods of high volume or bursts.
- *Browser, OS, and machine independent.* The software must work well with all major browsers, operating systems, machines, and connection bandwidths, as can be expected in

the heterogeneous population of users.

- *Adaptive.* The recognition or other core software should learn rapidly to quickly evaluate the quality of submitted data and permit timely feedback to contributors.

Open data collection serves as the core of the Open Mind Initiative (www.OpenMind.org), which currently has three projects: handwriting recognition, speech recognition, and a common-sense reasoning system. Software developers who contribute to the Initiative explore and develop the new software methodologies I've described.

Open data collection, as outlined, is not a panacea; it will not eliminate the need for careful analysis and research. It can be used in conjunction with other techniques such as traditional data mining. Open data collection offers

a promising approach to further promote the development of intelligent systems. In the highly networked, open collaborative environment provided by the Web, the data needs of intelligent systems development drive the search for large data sets and new software technologies. The software technologies that emerge from this novel approach to creating intelligent software may well prove useful across our entire field. *

David G. Stork is the chief scientist at Ricoh Silicon Valley. Contact him at stork@rsu.ricoh.com.

Editor Info: Michael J. Lutz, Rochester Institute of Technology, Department of Computer Science, 102 Lomb Memorial Drive, Rochester NY 14623; (phone) +1 716 465-2909; (fax) +1 716 475 7100; mjl@cs.rit.edu

15th IEEE SYMPOSIUM on COMPUTER ARITHMETIC



ARITH 15
June 11-13, 2001
Vail, Colorado

CALL FOR PAPERS

Authors are invited to submit papers describing recent advances on all aspects of computer arithmetic, including, but not restricted to the following topics:

- Foundations of number systems and arithmetic;
- Arithmetic processor design and implementation;
- Arithmetic algorithms and their analysis;
- Highly parallel arithmetic units and systems;
- New floating-point units and systems;
- Standards for number representation and arithmetic;
- Impact of high level languages on arithmetic systems;
- Special function implementation; and
- Applications of Computer Arithmetic for cryptography, DSP, data compression and others.

Postscript version of the complete paper, and a separate text file with title and abstract, should be submitted by **no later than November 1, 2000** following the instructions in the conference web site:

<http://ar15.polito.it/>

