SOLVING CROSSWORD PUZZLES VIA THE GOOGLE API

David E. Goldschmidt
Rensselaer Polytechnic Institute
Troy, New York, USA
goldsd@cs.rpi.edu

Mukkai Krishnamoorthy
Rensselaer Polytechnic Institute
Troy, New York, USA
moorthy@cs.rpi.edu

ABSTRACT

The Google™ API enables software agents to query and use search results from the large collections of data available via the ever-popular Google search engine. Web searches using Google are exposed to over 4 billion pages, many of which are cached within Google. While the Google API may be used to produce customized user interfaces to Google, the API also provides direct programmatic access to the subset of the Web covered by Google. In this paper, we present a fresh approach to solving crossword puzzles by making use of the Google API. Our system, the *Google CruciVerbalist* (*GCV*), reads XML-encoded crossword puzzles, derives answers to clues via the Google API, and uses a refined depth-first search algorithm to populate the crossword grid. GCV has successfully solved smaller puzzles, especially ones containing pop-culture and fill-in-the-blank types of clues. Based on this ongoing work, limitations of current search technologies are identified. To overcome these limitations, we look ahead to semantic queries via the emerging *Semantic Web*, including techniques using RDF that augment the Google search engine with semantic information, enabling semantically rich queries beyond the current capabilities of Google.

KEYWORDS

Crossword Puzzles, Google, RDF, Searching, Semantic Web, XML

1. INTRODUCTION

In this paper, we present an approach to solving crossword puzzles that uses the Google API to obtain answers to puzzle clues from Google's vast and ever-growing knowledge base. A crossword puzzle consists of an empty square grid and a set of clues keyed to the grid by both number and direction (see Figures 2 and 3 for examples).

While the precursor problem of constructing a crossword puzzle (i.e. filling an empty crossword grid with words from a given dictionary) has been sufficiently addressed (Ginsberg et al 1990), solutions to the adjunct problem of actually solving a crossword puzzle via software are few and far between. The most (and perhaps only) sophisticated system thus far is PROVERB, developed at Duke University (Keim et al 1999). Though PROVERB boasts impressive success rates, including an average of 95% words correct over a test set of 370 crossword puzzles, the system has an excessively large search space (including all combinations of letters) and requires a great deal of computing horsepower. Further, it uses a crossword database (CWDB) of approximately 350,000 clue-target pairs built from 5142 previously solved puzzles. Use of such a database is effective albeit too domain-specific and not generally reusable in other contexts.

We applied our simplified Google-only approach to solving crosswords to a variety of puzzles, ranging in difficulty from the classic *New York Times* and *Los Angeles Times* puzzles to the pop-culture *TV Guide* puzzles to puzzles designed for children. As a work in progress, GCV has successfully solved smaller crossword puzzles in which many clues are fill-in-the-blank style clues related to pop-culture (e.g. *TV Guide* or *People* crosswords). As work continues on the refined depth-first search algorithm, solutions to *TV Guide* and *People* puzzles should result.

2. SOLVING CROSSWORD PUZZLES

Making use of the Java-based Google API to programmatically query Google, we have built a lightweight Java application that solves crossword puzzles. The prototype, as shown in Figure 1, is called the *Google CruciVerbalist* (*GCV*) and has the flexibility to translate crossword puzzles of all shapes and sizes via a generic XML interface. Once translated, GCV sends clues to Google and scans results for candidate answers, ranking such answers based on word frequency and relative context.

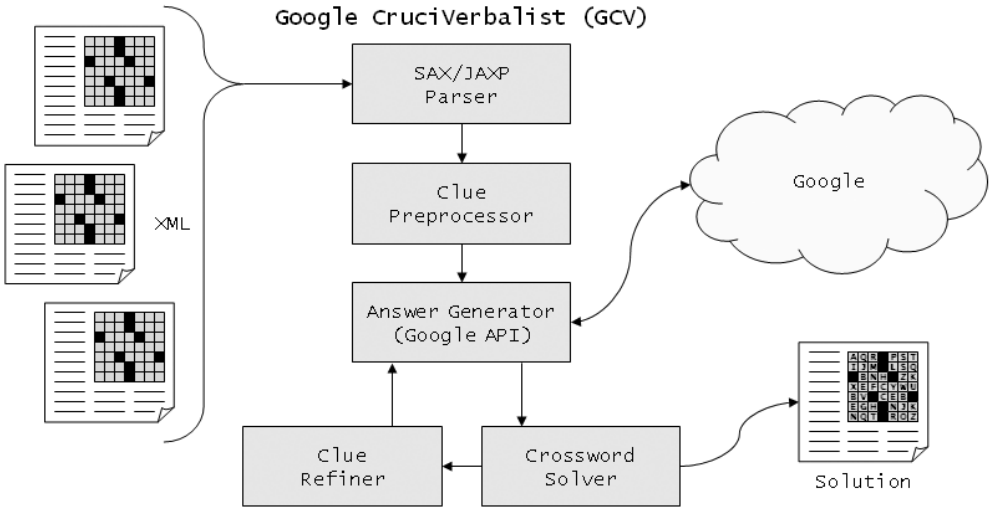

Figure 1. Architectural diagram of the Google CruciVerbalist (GCV)

Using a modified depth-first search algorithm, GCV attempts to fill in the crossword grid while sending new queries back out to Google to refine results and confirm answers. In addition to assigning probabilistic weights to candidate answers, GCV assigns weights to positions within the crossword grid, thus a dense portion of the grid that's substantially filled in will likely remain intact.

As noted in Keim et al 1999, the task of solving a crossword puzzle is twofold: (1) deciphering and answering natural language questions that require a broad but shallow knowledge base; and (2) populating the crossword grid with such answers in an attempt to maximize the number of correctly identified words. These two tasks are very much intertwined.

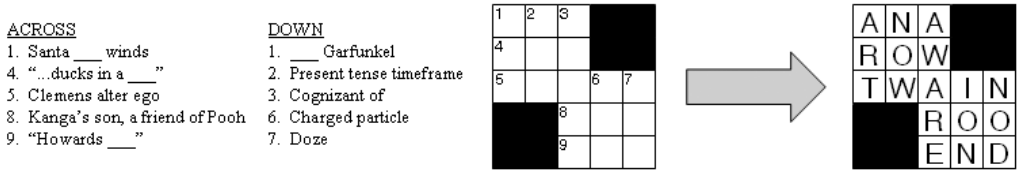

Figure 2. A 5x5 sample crossword puzzle that GCV has successfully solved

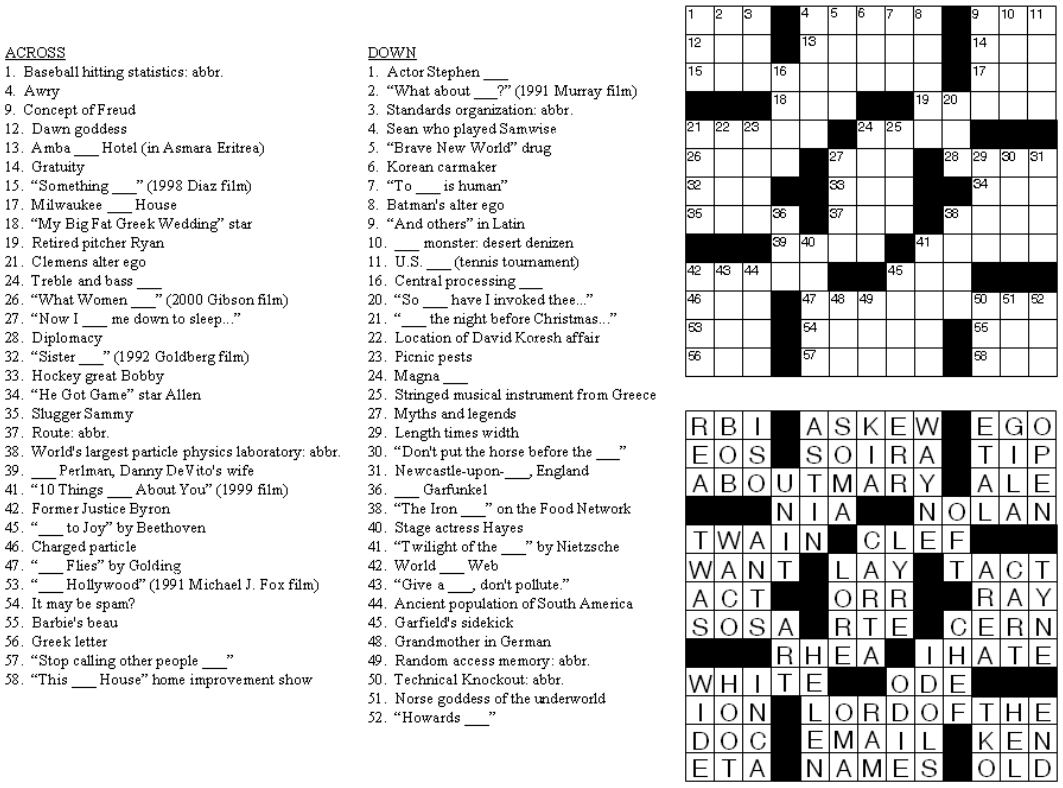

Figure 3. A 13x13 sample crossword puzzle that GCV has successfully solved

2.1 XML Input Description

Crossword puzzles are represented via a simple XML document, an example of which is shown below for the crossword puzzle in Figure 2. GCV inputs the XML crossword puzzle via the *Simple API for XML* (*SAX*) and the *Java API for XML Processing* (*JAXP*), the de facto approach to reading and interpreting XML documents in Java (see Niemeyer and Knudsen 2002).

```
<?xml version="1.0" encoding="UTF-8"?>
<Puzzle type="crossword">
 <Title>Test Puzzle 2</Title>
 <Author>David Goldschmidt</Author>
 <Dimensions>5,5</Dimensions>
 <Clue direction="across" ID="1" length="3" question="Santa ___ winds" answer="ana" />
 <Clue direction="across" ID="4" length="3" question="&amp;quot;...ducks in a ___&amp;quot;" answer="row" />
 <Clue direction="across" ID="5" length="5" question="Clemens alter ego" answer="twain" />
 <Clue direction="across" ID="8" length="3" question="Kanga&amp;apos;s son, a friend of Pooh" answer="roo"/>
 <Clue direction="across" ID="9" length="3" question="Howards ___" answer="end" />
 <Clue direction="down" ID="1" length="3" question="___ Garfunkel" answer="art" />
 <Clue direction="down" ID="2" length="3" question="Present tense timeframe" answer="now" />
 <Clue direction="down" ID="3" length="5" question="Cognizant of" answer="aware" />
 <Clue direction="down" ID="6" length="3" question="Charged particle" answer="ion" />
 <Clue direction="down" ID="7" length="3" question="Doze" answer="nod" />
 <NumberLocation ID="1" location="0,0" />
 <NumberLocation ID="2" location="1,0" />
 <NumberLocation ID="3" location="2,0" />
 <NumberLocation ID="4" location="0,1" />
 <NumberLocation ID="5" location="0,2" />
 <NumberLocation ID="6" location="3,2" />
 <NumberLocation ID="7" location="4,2" />
 <NumberLocation ID="8" location="2,3" />
 <NumberLocation ID="9" location="2,4" />
</Puzzle>
```

2.2 Identifying Answers to Clues

The World Wide Web has clearly evolved into a vast knowledge base, including both a breadth and depth of innumerable topics. Since Google has proven itself to be one of the most comprehensive search engines, covering the breadth necessary for crossword puzzle clues, we use the Google API to suggest answers to such clues. The Google API provides direct programmatic access to send a query string to Google and obtain the results. Results are returned in sets of ten.

2.2.1 Preprocessing the Clues

Before we send clues to Google, we preprocess each clue in an attempt to maximize the "good" answers in the first ten query results. No additional result sets are requested during the initial pass. This keeps the list of potential answers small, reducing the overall search space. This also tends to avoid false positives from overwhelming the results set. Further, the Google API limits the number of queries one may process in a given day to 1000, which therefore serves as an upper bound on the allowable number of queries used to solve a crossword puzzle via GCV.

Preprocessing a clue involves such steps as removing extraneous punctuation (though we keep quotation marks intact), replacing blanks in fill-in-the-blank style clues with full-word wildcards, determining whether the number of words is evident from the clue, rearranging basic English grammar, and so on. Unless a clue has a full-word wildcard, clues are also duplicated and "synonyms of" is prepended to the clue. Table 1 provides examples of clues and in what forms they are sent to Google. These techniques help to increase the likelihood of identifying the correct answer. Further, sending multiple queries to Google for a single clue tends to reinforce a "good" answer since it will likely appear in each set of results.

Table 1. Transforming clues for Google. Clues are preprocessed and sent to Google in one or more altered forms

Original clue	Google-friendly clue(s) sent to Google	Answer
Mary ___ Little Lamb: 2 words	Mary * * Little Lamb "Mary * * Little Lamb"	*had a*
Diplomacy	Diplomacy synonyms of Diplomacy	*tact*
Intensive Care Unit (abbr.)	Intensive Care Unit synonyms of Intensive Care Unit abbreviations of Intensive Care Unit	*I. C. U.*
"Any Time ___" (Beatles tune)	"Any Time *" Beatles tune "Any Time * *" Beatles tune	*at all*
Type of dancing	* dancing * * dancing	*tap*
Superman's admirer	Superman's admirer admirer of Superman	*Lois*
Friends' Phoebe	*Friends'* Phoebe Phoebe of *Friends*	*Lisa*
Not dry	Not dry opposite of dry antonyms of dry	*wet*
Knight or Danson	Knight or Danson Knight Danson	*Ted*

2.2.2 Postprocessing the Google API Results

As results are retrieved, a simple concordance is associated with each clue. Candidate answers are selected out of the title and snippets that Google returns. Since we know the target length q of the correct answer (based on the crossword grid), we record each word within the results set of length q. Further, each set of

consecutive words that together are length *q* is recorded as a "multiword," since such multiwords are often used in crosswords.

Associated with each word is the number of occurrences that word has within the results. After all result sets are processed for a given clue, a confidence value *c* is calculated for each word. The confidence value *c* is simply the number of occurrences of a given word divided by the overall number of occurrences of all words. Thus, words that appear many times in the result sets have a higher confidence value.

In some cases (e.g. crosswords for children), we know from the clue the number of words in the answer. Once we detect that the number of words is specified in a given clue (e.g. "Cool ___ cucumber: 2 words"), we also know that all clues that do not have such a specification have 1-word answers. This knowledge helps to reduce the resulting search space.

2.3 Populating the Crossword Grid

After a set of potential answers has been identified for each of the clues of the crossword puzzle, we attempt to fill the crossword grid with as many correct words as possible. A straightforward strategy is to fill in words that have the highest confidence values first, backtracking as necessary when dead-ends are encountered. As each word is placed, the word lists of the adjacent clues would be reduced accordingly. If placing a word reduces an adjacent word list altogether, we may then decide to backtrack.

Aside from the need of excessive computing horsepower, this traditional depth-first search approach provides no guarantee of a solution, since the Google results are certainly not guaranteed to contain correct answers. To address this problem, we relax the "exact fit" requirement, allowing high-confidence words to be placed even when perpendicular word possibilities do not entirely exist. We also integrate additional Google queries during grid population to obtain more results from Google to help locate high-confidence words that do fit.

Rather than work in an enormous search space, our approach is to begin with a potentially incomplete search space and attempt to fill in the low-confidence gaps by expanding the search space as we progress.

3. RESULTS

GCV has been applied to a myriad of crossword puzzles, including daily puzzles from *The New York Times* and *Los Angeles Times*, pop-culture puzzles from *TV Guide*, and educational puzzles designed for children ages 7 and up. GCV excels at correctly answering fill-in-the-blank style and pop-culture clues, these being the first answers populated in the crossword grid during the solving process.

The crossword puzzle shown in Figure 2 was solved using 20 distinct Google queries; of the 10 clues, 6 (60%) result in the correct answer occurring most frequently in the Google results. Likewise, the crossword puzzle shown in Figure 3 was solved using 160 distinct Google queries; of the 68 clues, 54 (79.4%) result in the correct answer occurring most frequently. In puzzles that GCV was unable to completely solve, approximately 30-50% of the clues result in the correct answer ranked highest in GCV.

One such 13x13 puzzle from *TV Guide* that GCV was unable to completely solve consisted of 68 clues. GCV placed 43 answers, of which 40 were correct (58.8%). Of the 135 total squares of the crossword grid, 102 were populated with the correct letter (75.6%). As work continues on the refined depth-first search algorithm, solutions to this and other such puzzles should result.

Somewhat surprisingly, GCV has had limited success in solving puzzles designed for children. Clues that are straightforward even to a child can often be a challenge to Google, indicating that keyword-based searching is not the best approach.

4. THE EMERGING SEMANTIC WEB

For many, a search engine is the starting point for locating new information and services on the Web. Google's search technologies rely on the linked structure of the Web to rank webpages based on their popularity. Other search engines use a variety of word-frequency and clustering techniques, as well as

additional approaches. Regardless of the underlying architecture, users specify keywords that match words in huge search engine databases, producing an ordered list of webpage addresses (i.e. URLs) and snippets of webpages in which the keywords matched.

While such technologies have been successful, users are still often faced with the daunting task of sifting through multiple pages of results, many of which are irrelevant. Surveys indicate that almost 25% of Web searchers are unable to find useful results in the first set of URLs that are returned (Roush 2004). Sophisticated as search engines have become, they are still often unable to bridge the gap between human understanding and HTML data.

Tim Berners-Lee, the inventor of the World Wide Web, defines the *Semantic Web* simply as "The Web of data with meaning in the sense that a computer program can learn enough about what the data means to process it." (Berners-Lee 1999) Rather than a Web filled only with human-interpretable information, Berners-Lee's vision includes an extended Web that incorporates *machine-interpretable information*, thus enabling machines to process and make sense of the volumes of available information, acting on behalf of their human counterparts. In Berners-Lee's words, "This raises the exciting possibility of letting programs run over this material and help us analyze and manage what we are doing....to help us deal with the bulk of data, to take over the tedium of anything that can be reduced to a rational process, and to manage the scale of our human systems." (Fensel et al 2003)

4.1 Building Blocks of the Semantic Web

Much of the infrastructure of the Semantic Web has already been defined (see Berners-Lee et al 2001, Fensel et al 2003, Hjelm 2001, Connolly et al 2001 and 2003, Heflin et al 2002, Lassila and Swick 1999, and so on).

The *eXtensible Markup Language* (*XML*) provides a Web-friendly means for representing data, omitting aspects of presentation (see Castro 2001, Schmelzer 2002). XML is a powerful text-based markup language that, when coupled with syntactical specifications (e.g. *XML Schema*), enables users to define arbitrarily complex data exchange formats. Though many applications tout support for XML, interoperability between heterogeneous applications is often severely limited, since XML documents lack machine-interpretable semantic information. Nonetheless, XML serves as a foundation for other Semantic Web building blocks.

While most are familiar with the *Uniform Resource Locator* (*URL*) as an address on the Web, the more generic *Uniform Resource Identifier* (*URI*) is not as well-known. URIs are used to represent tangible objects, people, places, abstract relationships, intangible or fuzzy concepts—just about anything (Connolly 2003). Syntactically, URIs resemble URLs. As an example, the URI `http://www.cs.rpi.edu/~goldsd` may be used to represent David Goldschmidt, one of the authors of this document. The URI for the State of New York in the United States of America may be `http://www.state.ny.us/`.

Defining URIs enables the development of ever-expanding machine-interpretable vocabularies. These are the nouns, verbs, and other language constructs that make up the Semantic Web. The *Resource Description Framework* (*RDF*) is used to combine URIs together to form machine-interpretable sentences. Akin to simple prose, an RDF statement consists of a subject, a predicate, and an object. In general, the subject is a resource, the predicate is a property, and the object is either a resource or a literal value (see Lassila and Swick 1999, Manola and Miller 2002, Hjelm 2001, Powers 2003).

Since RDF is a framework, it sometimes lacks immediate application on its own; however, using basic RDF constructs, rich machine-interpretable semantic vocabularies may be developed. *RDF Schema* (*RDFS*) and the *DARPA Agent Markup Language + Ontology Inference Language* (*DAML+OIL*) are examples of widely used vocabulary languages based on RDF. RDF Schema allows you to define classes, subclasses, and properties (see Brickley and Guha 2002). DAML+OIL provides a richer semantic vocabulary, including such constructs as: (1) equivalence relations; (2) cardinality specifications; (3) basic set relationships; (4) an inverse relation; and (5) enumerations (Connolly 2001, Heflin 2002).

Using RDF Schema and DAML+OIL, domains of knowledge—called *ontologies*—are defined. An ontology "formally defines a common set of terms that are used to describe and represent a domain," thus making the terms and knowledge therein reusable (Fensel et al 2003).

4.2 Crossword Puzzles and the Semantic Web

Currently, GCV is sometimes unable to obtain useful candidate answers from Google. For example, the clue "A canary or a kitten" (answer: *pet*) stumped GCV's initial Google search. In a Semantic Web environment, we may find via a zoological ontology that a canary is a species of bird, which in turn may be a pet; likewise, a kitten is a kind of cat, which may also be a pet (see Figure 4).

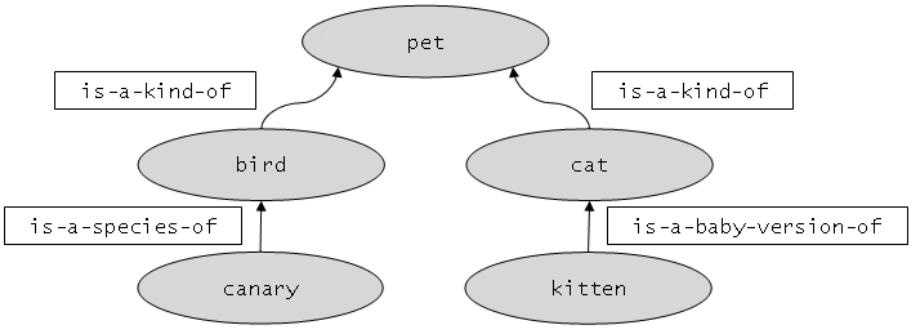

Figure 4. A segment of a potential zoological ontology defining both *resources* (ovals) and *properties* (rectangles)

As another example, consider the clue "Clemens alter ego" (answer: *Twain*). As shown in Figure 5, a search of the Semantic Web may yield such resources as Roger Clemens and Samuel Clemens, and the *has-alter-ego* property. To infer the correct answer, we traverse the web of properties equivalent to *has-alter-ego* and find that the *also-known-as* property leads from Samuel Clemens to Mark Twain.

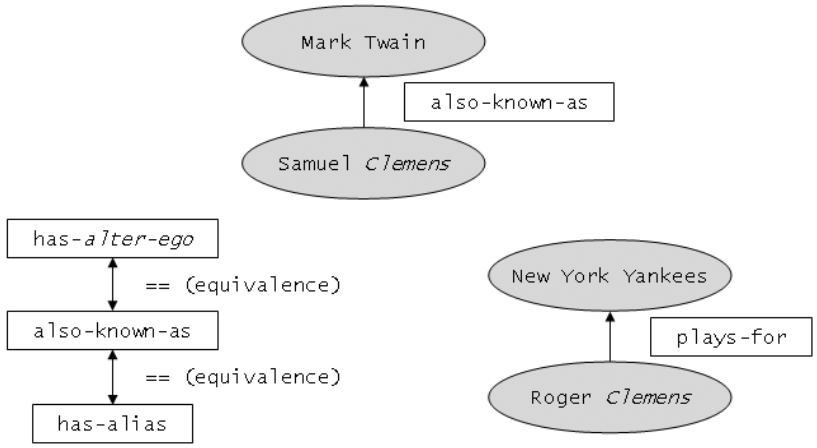

Figure 5. Potential Semantic Web search results for the clue "Clemens alter ego" with simple keyword matches italicized. Note that ovals represent *resources* and rectangles represent *properties*

These (and other) examples reveal the flexibility and extensibility of the Semantic Web. Instead of forcing a user to sift through irrelevant search results, the Semantic Web enables machines to infer pertinent results from semantically structured information. Detecting and traversing such relationships via ontologies is a far more effective approach to mimicking simple human thought patterns than simple keyword-based searching.

An additional attribute of the Semantic Web architecture that is easily overlooked is the Web-based representation of properties. Unlike the traditional object-oriented approach in which property (i.e. attribute and relationship) definitions are defined *within* a class definition, properties in the Semantic Web are separate, traversable networks. Removing the direct interdependency between resources and properties increases the ability to share, standardize, reuse, and exchange property definitions.

5. CONCLUSION

The World Wide Web has grown tremendously since its inception in the early 1990s, evolving from pockets of simple hand-edited HTML files to a wide range of elaborate machine-generated websites. The Semantic Web is an inevitable next step in this evolution, enabling machines to interpret and better utilize the multitudes of information already available on the World Wide Web.

Key to the Semantic Web's success is an efficient means for obtaining information from the existing Web and transforming it into machine-interpretable information for the Semantic Web. As GCV solves crossword puzzles, it acquires knowledge in the form of clue-answer pairs. Such knowledge may be captured and represented in RDF, ultimately forming a crossword puzzle ontology. Given the broad nature of crossword puzzle clues, the resulting crossword puzzle domain should prove both interesting and useful to a search engine for the Semantic Web.

Research and continued work on the Semantic Web will ultimately create an additional metadata layer atop the existing World Wide Web. It is this metadata layer that machines will be able to process and interpret on behalf of their human counterparts, increasing the relevance and applicability of Web-based information and services.

REFERENCES

Berners-Lee, T., 1999. *Weaving the Web: the Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. HarperSanFrancisco, New York, New York, USA.

Berners-Lee, T. et al, 2001. The Semantic Web. *Scientific American*. May 2001. http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html.

Brickley, D. and Guha, R., editors, April 30, 2002. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C, http://www.w3.org/TR/rdf-schema.

Calishain, T. and Dornfest, R., 2003. *Google Hacks: 101 Industrial-Strength Tips & Tools*. O'Reilly & Associates, Inc., Sebastopol, California, USA.

Castro, E., 2001. *XML for the World Wide Web: Visual QuickStart Guide*. Peachpit Press, Berkeley, California, USA.

Connolly, D. et al, December 18, 2001. *Annotated DAML+OIL Ontology Markup*. W3C, http://www.w3.org/TR/daml+oil-walkthru.

Connolly, D. et al, December 18, 2001. *DAML+OIL (March 2001) Reference Description*. W3C, http://www.w3.org/TR/daml+oil-reference.

Connolly, D. et al, October 23, 2003. *Web Naming and Addressing Overview (URIs, URLs, ...)*. W3C, http://www.w3.org/Addressing.

Fensel, D. et al, editors, 2003. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, Cambridge, Massachusetts, USA.

Ginsberg, M. et al, 1990. Search lessons learned from crossword puzzles. In *Proceedings of the Eighth National Conference on Artificial Intelligence*. USA, pp. 210-215.

Heflin, J. et al, editors, July 8, 2002. *Requirements for a Web Ontology Language*. W3C, http://www.w3.org/TR/webont-req.

Hjelm, J., 2001. *Creating the Semantic Web with RDF*. John Wiley & Sons, Inc., USA.

Keim, G. et al, 1999. PROVERB: the probabilistic cruciverbalist. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. USA, pp. 710-717.

Lassila, O. and Swick, R., editors, February 22, 1999. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C, http://www.w3.org/TR/REC-rdf-syntax.

Manola, F. and Miller, E., editors, April 26, 2002. *RDF Primer*. W3C, http://www.w3.org/TR/rdf-primer.

Niemeyer, P. and Knudsen, J., 2002. *Learning Java™, Second Edition*. O'Reilly & Associates, Inc., Sebastopol, California, USA.

Powers, S., 2003. *Practical RDF*. O'Reilly & Associates, Inc., Sebastopol, California, USA.

Roush, W., 2004. Search beyond Google. *Technology Review*. March 2004. http://www.technologyreview.com/articles/print_version/roush0304.asp.

Schmelzer, R. et al, 2002. *XML and Web Services Unleashed*. Sams Publishing, USA.