Searching for Common Sense: Populating Cyc™ from the Web

**Cynthia Matuszek, Michael Witbrock, Robert C. Kahlert,
John Cabral, Dave Schneider, Purvesh Shah, Doug Lenat**
Cycorp, Inc.
3721 Executive Center Drive, Suite 100, Austin, TX 78731
{cynthia, witbrock, rck, jcabral, daves, shah, lenat}@cyc.com

Abstract

The Cyc project is predicated on the idea that effective machine learning depends on having a core of knowledge that provides a context for novel learned information – what is known informally as "common sense." Over the last twenty years, a sufficient core of common sense knowledge has been entered into Cyc to allow it to begin effectively and flexibly supporting its most important task: increasing its own store of world knowledge. In this paper, we present initial work on a method of using a combination of Cyc and the World Wide Web, accessed via Google, to assist in entering knowledge into Cyc. The long-term goal is automating the process of building a consistent, formalized representation of the world in the Cyc knowledge base via machine learning. We present preliminary results of this work and describe how we expect the knowledge acquisition process to become more accurate, faster, and more automated in the future.

1 Introduction

The idea of building a very large-scale knowledge base that can be used as a foundation for automated knowledge acquisition has been present in artificial intelligence research for more than twenty years [Lenat *et al.,* 1983]. In that time, an enormous amount of progress has been made [Thrun *et al.*, 1998]; techniques developed under the umbrella of machine learning have been successfully applied to work ranging from robotics, to voice recognition, to bioinformatics. In all of these fields, the use of preexisting knowledge is widespread. Much of this work relies on either programming an inductive bias into a learning system (*e.g.*, in systems like AM [Lenat, 1976]); or on providing an inductive bias in the form of training examples [Brown, 1996].

Also in that time, the Web has emerged as a huge repository of electronically available knowledge, and indexing systems such as Google have made that knowledge progressively more accessible [Brin and Page, 1998]. Work that relies on the web in general, and Google in particular, for information extraction is proving to be a fertile research area [Ghani, 2000; Kwok *et al.* 2001; Etzioni *et al.* 2004].

The purpose of the Cyc project is to provide computers with a store of formally represented "common sense": real world knowledge that can provide a basis for additional knowledge to be gathered and interpreted automatically [Lenat, 1995]. In the last twenty years, over three million facts and rules have been formally represented in the Cyc knowledge base by ontologists skilled in CycL, Cyc's formal representation language. Tools have been developed which allow subject matter experts to contribute directly [Panton *et al.*, 2002; Witbrock *et al.*, 2003; Belasco *et al.*, 2004]. In addition, natural language generation and parsing capabilities have been developed to provide support for learning from English corpora [Witbrock *et al.*, 2004].

As a result, the Cyc knowledge base now contains enough knowledge to support experimentation with the acquisition of additional knowledge via machine learning. In this paper, we describe a method for gathering and verifying facts from the World Wide Web. The knowledge acquisition procedure is described at both an overview level and in detail. The work focuses on three novel approaches: using knowledge already in the Cyc KB to focus the acquisition of further knowledge; representing acquired knowledge in the knowledge base; and using Google in two distinct ways, to find facts and, separately, to verify them.

While this research is at an early stage, the initial results are promising in terms of both the acquisition speed and quality of results. Even in its preliminary form, the mechanism described is a useful tool for reducing the cost of manually entering knowledge into Cyc; the level of expertise required to enable a person to contribute to the KB is reduced, and many of the necessary steps are handled automatically, reducing the total time required. The number of sentences that can be acquired in this way and reviewed for accuracy by an untrained reviewer outstrips the rate at which sentences can be hand-authored by a trained ontologist, and the verification steps reduce the amount of work required of a human reviewer by approximately 90%.

2 Cyc and CycL

The Cyc system is made up of three distinct components, all of which are crucial to the machine learning process: the knowledge base (KB), the inference engine, and the natural language system. The Cyc KB contains more than 3.2 million assertions (facts and rules) describing more than 280,000 concepts, including more than 12,000 conceptinterrelating predicates. Facts stored in the Cyc KB may be atomic (making them Ground Atomic Formulæ, or *GAFs*), or they may be complex sentences. Depending on the predicate used, a GAF can describe instance-level or type-level knowledge. All information in the KB is asserted into a hierarchical graph of *microtheories*, or reasoning contexts [Guha, 1991; Lenat, 1998].

CycL *queries* are syntactically legal CycL sentences, which may be partially bound (that is, contain one or more variables). The Cyc inference engine is responsible for using information in the KB to determine the truth of a sentence and, if necessary, find provably correct variable bindings.

Sample instance-level GAF:

(foundingDate Cyc (YearFn 1985))

Sample entity-to-type and type-to-type GAFS:

(sellsProductType SaudiAramco PetroleumProduct)

(conditionAffectsPartType CutaneousAnthrax Skin)

Sample non-atomic sentence:

```
(or   (foundingDate AlQaida (YearFn 1987)))
      (foundingDate AlQaida (YearFn 1988)))
```

Sample query:

(foundingAgent PalestineIslamicJihad *?WHO*)

The natural language component of the system consists of a lexicon, and parsing and generation subsystems. The lexicon is a component of the knowledge base that maps words and phrases to Cyc concepts, while various parsers provide methods for translating English text into CycL. The system also has a relatively complete ability to render CycL sentences into English, although the phrasing can be somewhat stilted when longer sentences are generated.

The work described in this paper targets the automatic acquisition of GAFs. Simple facts are more likely to be readily found on the web, and this approach minimizes difficulties in generating and parsing complex natural language constructs.

2.1 Overview of the Learning Cycle

Gathering information from the web proceeds in six stages, as illustrated in Figure 1:

1. Choosing a query: Because the number of concepts in the KB is so large, the number of possible CycL queries is enormous; choosing interesting, productive queries automatically is a necessary step in automating the knowledge acquisition process. An example of such a query might be:

(foundingAgent PalestineIslamicJihad *?WHO*)

2. Searching: Once a query is selected, it is translated into one or more English search strings. The query above might be rendered into strings such as:

"PIJ, founded by "

"Palestine Islamic Jihad founder "

These strings are passed on to the Google API. The appropriate sections of any resulting documents are downloaded, and the relevant section is extracted (*e.g.*, "PIJ founder Bashir Musa Mohammed Nafi is still at large…").

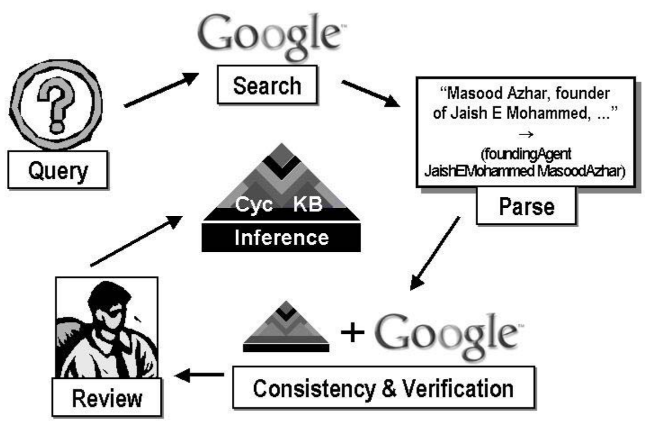

Google
Search
"Masood Azhar, founder of Jaish E Mohammed, ..." → (foundingAgent JaishElMohammed MasoodAzhar)
Parse
Query
Cyc KB
Inference
+ Google
Consistency & Verification
Review

Figure 1: Learning is a process of selecting interesting questions, searching for that information on the web, parsing the results, performing verification and consistency checks with the document corpus and the KB, reviewing, and asserting that knowledge into the KB.

3. Parsing results: The relevant components of sentences are identified by their location relative to the search string. The terms are then parsed into CycL via the natural language parsing process described in section 3.3, resulting in one or more GAFs such as:

(foundingAgent PalestineIslamicJihad Terrorist-Nafi)

4. KB consistency checking: Some of the results retrieved during the search process are disprovable, because they are inconsistent with knowledge already present in the knowledge base; others are already known or trivially provable, and therefore redundant. Any GAF found via inference to be inconsistent or redundant is discarded.

5. Google verification: During search, the largest possible set of candidate GAFs is created. Those GAFs that are not discarded during KB consistency checking are re-rendered into English search strings, such as:

"Bashir Nafi is a founder of Palestine Islamic Jihad"

and a second Google search over those strings is performed. Any GAF that results in no retrieved documents during this phase is discarded.

6. Reviewing and asserting: The remaining GAFs are asserted into special hypothetical contexts in the knowledge base. An ontologist or human volunteer reviews them for accuracy, using a tool specific to that task [Witbrock *et al.*, 2005], and the ones found to be correct are asserted into the knowledge base.

3 Implementation of the Learning Cycle

3.1 Selecting Queries

While it is often useful in an application context to look for the answer to a specific question, or automatically populate a class of information (such as founders of groups, or prime ministers of countries), satisfying the ultimate goal of populating the Cyc KB via machine learning relies in part on automatically selecting suitable sentences. There are a number of challenges that must be met in this regard. Queries should have reasonable probability of having interesting bindings and of being findable in the corpus (in this case, on the web). Some searches are unlikely to be productive for semantic reasons: some argument positions are of an infinite or continuous type, such as Time-Quantity, and could therefore yield an infinite number of mostly-uninteresting searches, such as (age *?OBJECT* (YearsDuration 300)). Other queries are guaranteed to be redundant.

We initially limited searches to a set of 134 binary predicates which, when used to generate search strings, tended to maximize useful results from web searches.[1] The algorithm for proceeding through those predicates was as follows:

For a given search run, a depth of D is selected. D is the maximum number of different values that can be used for each argument of a predicate. For each binary predicate p_i in our test set P (where $|P| = 134$), we retrieve from the KB the type constraints on each of its two arguments. Unless the type generalizes to an infinite class, we retrieve the D most fully represented values from the knowledge base – that is, those that appear in the most assertions, and therefore about which the most is known. These are assumed to be the most interesting terms of that type, and therefore the ones most likely to be found by a web search. For p_i we then have types T^{i1} and T^{i2}. The D best represented values would be (t^{i1}_1 … t^{i1}_D) and (t^{i2}_1 …t^{12}_D). If neither of a predicate's arguments took values of a continuous type, there would be $2D*|P|$ queries generated:

(p_1 t^{11}_1 ?VAR) … (p_1 t^{11}_D ?VAR)
(p_1 ?VAR t^{12}_1) … (p_1 ?VAR t^{12}_D)
…
($p_{|P|}$ $t^{|P|1}_1$?VAR) … ($p_{|P|}$ $t^{|P|1}_D$?VAR)
($p_{|P|}$?VAR $t^{|P|2}_1$) … ($p_{|P|}$?VAR $t^{|P|2}_D$)

For example, a set of the predicates foundingAgent and foundingDate, given a depth of 1, would produce three queries:

(foundingAgent AlQaida *?WHO*)
(foundingAgent *?WHAT* Terrorist-Salamat)
(foundingDate AlQaida *?WHEN*)

The fourth permutation is not produced, because the argument constraint, Date, is of a continuous type.

This approach is not without problems. It relies heavily on the nature of the type constraints placed on predicates; for some predicates, such as foundingDate, this works well, while for others the argument constraints are too broad. For example, the predicate sellsProductType takes a constant of type somethingExisting in its second argument position, because almost anything *can* be sold. The proposed way to address this problem is with a predicate typicalArgIsa, which would connect predicates such as sellsProductType with the collections to which they *typically* refer (in this case, CommodityProduct)[2].

Another problem lies with the assumption that the members of a class about which Cyc knows the most are the most interesting ones, which is only sometimes correct. The best-described instances of the class Person, for example, tend to be the ontologists who work at Cycorp, rather than, for example, heads of state. Future work will involve using Google in the selection of appropriate queries. Rather than using the top D most supported terms in the KB of type T, it should be possible to retrieve the hit count for up to several thousand members of T, and seek information about the D terms of T for which the most information is available.

3.2 Search

Generating Search Strings

Once the system has selected a query, it generates a series of search strings. The existing NL generation machinery is applied to 233 manually created special generation templates for the 134 predicates. [3] Several factors motivated the construction of specialized search generation templates within the NL system. In addition to the fact that productive search strings are often unlike standard English, CycL generations tend to be somewhat stilted, since ontologists have preferred unambiguous expression of CycL meanings over naturalness. In addition, the KB generally contains one or two generation templates for any given predicate, while there may be many common ways of expressing the information that may be useful for searching.

For example, for the query (foundingAgent PalestineIslamicJihad ?X), the system generates the following set of search strings:

Palestine Islamic Jihad founder ____
Palestinian Islamic Jihad founder ____
PIJ founder ____
Palestine Islamic Jihad, founded by ____
Palestinian Islamic Jihad, founded by ____
PIJ, founded by ____

All possible searches were generated from the Cartesian product of the generation templates with all English renderings of the arguments. In this example, Cyc knows three names for PalestineIslamicJihad, and has two templates for foundingAgent, resulting in six search strings. To simplify match detection, argument positions were only allowed at the beginning or end of the template string. In order to carry out the actual search, the "___" placeholders were stripped

[1] Examples: foundingAgent, foundingDate, sellsProductType, primeMinister, lifeExpectancy, awardWinners. Productive predicates were found via manual trial and error, from a set of domains selected to span a broad portion of the KB (terrorism, medical technology, conceptual works, global politics, family relationships, and sales).

[2] In most cases, these constraints will be learned from analysis of common usage in the existing corpus.

[3] Slightly fewer than half of the predicates have only one associated search template. Many obvious templates exist but have not been represented; in future work, an analysis of the search strings will be performed to determine what types of strings produce good results. That information will be used for automatically generating search strings for other predicates.

off, and the remaining string was submitted as a quoted string to Google.

Searching via Google
The search string is used with an interface to the Google API to retrieve a tuple consisting of the URL, the Google ranking, the match position and the web page text, which is then handed off to a parser that attempts to convert it into a meaningful CycL entity.

3.3 Parsing into CycL

Once a document is retrieved, the answer must be found and interpreted. First, the system searches for the exact text of the query string, and returns the search string plus either the beginning or the end of the sentence, depending on the position of the "___" in the generated string. The resulting string is searched for phrases that can be interpreted as a CycL concept that meets the type constraints imposed by the predicate. For example, in the string "PIJ founder Bashir Musa Mohammed Nafi is still …", "Bashir Musa Mohammed Nafi" is recognized as a person, and therefore as a candidate for the arg2 position of foundingAgent. For predicates that require strings (such as nameString, which relates an entity to its name), a named entity recognizer [Prager *et al.,* 2000] is used to find a suitable candidate.

In other cases, standard parsing techniques are used to try to find a useful interpretation, including looking up strings in Cyc lexicon, interpreting them as noun compounds or dates, and compositional interpretation. For speed, compositional interpretation is only attempted for terms judged to be constituents by a probabilistic CFG parser [Charniak, 2001]. This has the effect of eliminating a few correct answers (mostly where the parser produces an incorrect syntactic parse), but also decreases the total time spent on analysis by at least 50%.

Creating candidate CycL Sentences
The result of parsing the matched section in the web page is a set of candidate CycL terms, usually constants such as Terrorist-Nafi. Substituting these terms into the original incomplete CycL query produces a set of candidate GAFs.

3.4 Checking Cyc KB Consistency

In principle, any useful fact added to the system should neither disprovable nor trivially provable. For example, the sentence:

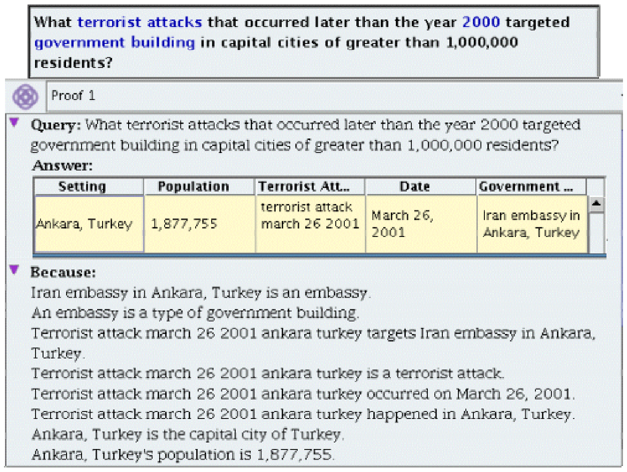
(foundingAgent PalestineIslamicJihad Terrorist-AlShikaki)

is not novel, because Cyc already knows this. Meanwhile,

(foundingAgent PalestineIslamicJihad AugusteRodin)

is novel, but trivially disprovable, as Cyc knows he died in 1917 (72 years before the PIJ was founded).

If the Cyc KB already contains knowledge that renders a new fact redundant or contradictory, that fact will be discarded. This is checked via inference; each new fact is treated as a query, and inference is performed to determine whether it can be proven false (inconsistent) or true (redundant.) Cyc provides justifications for facts used to produce query results, as in Figure 2, and it is helpful for redundant facts to be marked as additionally confirmed via search-based learning. Previous work suggests that one-step inference is sufficient to identify duplicate information or disprove contradictions during typical knowledge acquisition tasks [Panton *et al.,* 2002].


What terrorist attacks that occurred later than the year 2000 targeted government building in capital cities of greater than 1,000,000 residents?

Proof 1

Query: What terrorist attacks that occurred later than the year 2000 targeted government building in capital cities of greater than 1,000,000 residents?
Answer:

Setting	Population	Terrorist Att...	Date	Government ...
Ankara, Turkey	1,877,755	terrorist attack march 26 2001	March 26, 2001	Iran embassy in Ankara, Turkey

Because:
Iran embassy in Ankara, Turkey is an embassy.
An embassy is a type of government building.
Terrorist attack march 26 2001 ankara turkey targets Iran embassy in Ankara, Turkey.
Terrorist attack march 26 2001 ankara turkey is a terrorist attack.
Terrorist attack march 26 2001 ankara turkey occurred on March 26, 2001.
Terrorist attack march 26 2001 ankara turkey happened in Ankara, Turkey.
Ankara, Turkey is the capital city of Turkey.
Ankara, Turkey's population is 1,877,755.

Figure 2: Justifications for the claim that a 2001 attack on Ankara meets the criteria for the query being run.

3.5 Google Verification

In order to guard against parser errors and excessively general search terms (such as ambiguous acronyms), a second Google search is performed, in order to determine whether search strings generated from the new GAF will produce results. Search strings are generated from the candidate GAF that was learned, but any string containing an acronym or abbreviation (*e.g.,* "PIJ" for "Palestian Islamic Jihad") is supplemented with the disambiguation term: the least common word (based on Google hit counts) of the expanded acronym. In this case, the string "Palestine" is added as a term, since it is the least common word in the set 'Palestine,' 'Palestinian,' 'Islamic,' and 'Jihad.' The resulting verification search string is:

"PIJ founder Bashir Nafi" +"Palestine"

Any fact for which this verification step returns no results is considered unverified, and will not be presented to a reviewer.

3.6 Review and Assertion

In the final step, learned sentences are reviewed by a human curator, and, if correct, asserted into the Cyc KB. Currently, suggested sentences are presented to the reviewer in no particular order; in future, sorting methods will be implemented and tested. The most straightforward approaches involve making greater use of information already retrieved from Google: since information about the searches underlying a candidate sentence is stored in the KB, it should be possible and productive to give priority to sentences that are supported by a larger total number of documents. In effect, the numerical values returned during the verification step could be used to sort the most widely supported sentences upwards in the review process. Another possibility, if several contradictory facts are found, is giving review priority to those found in documents with the highest Google ranking.

4 Results

Statistics were gathered for a case in which 134 predicates in *P* were used, and *D* was set to 20.[4] The majority of the searches expended, about 80%, were performed in the verification phase rather than the initial search phase. The results were as follows:

Queries: 348
Searches expended: 4290 (817 initial, 3477 verification)
GAFs found: 1016
… and rejected due to KB inconsistency: 4
… and already known to the KB: 384
… and rejected by Google verification: 566
Novel GAFS found and verified: 61

A human reviewer then went through the verified GAFs, and a sample of 53 of the unverified GAFs, and determined their actual correctness rate. The results were as follows:

	Verified	Unverified
True (correct)	32	8**
False (incorrect)	29*	45

Total novel facts: 114
Novel, correct facts discovered: 77
Incorrect facts discovered: 37

Facts categorized correctly: 68%
Facts categorized incorrectly: 32%
… * false positives (false but verified): 25%
… ** false negatives (true but unverified): 7%

Examples of these result types:

Query: (#$hasBeliefSystems #$Iran ?X)

Search string: "Iran adheres to "

Candidate GAF: (#$hasBeliefSystems #$Iran #$Islam)

Verification search strings:
"Islamic Republic of Iran adheres to Islam"
"Iran adheres to Islam"
"Iran believes in Islam" *(found)*
"Islamic Republic of Iran believes in Islam" *(found)*

Example GAFs already known to the KB:
(#$vestedInterest #$Iran #$Iraq
(#$inhabitantTypes #$Lebanon #$EthnicGroupOfKurds)

Example GAFs rejected due to KB inconsistency:
(#$northOf #$Iran #$Iran)
(#$geopoliticalSubdivision #$Iraq #$Iran)

[4] It takes between four and five hours to exhaust an allotment of 3,000 searches per day through the Google API.

Correct, verified GAF:
(foundingDate AfricanNationalCongress (YearFn 1912))

***Incorrect but verified GAF:**
(foundingDate JewishDefenseLeague (DecadeFn 198))

Incorrect, rejected GAF:
(objectFoundInLocation KuKluxKlan GillianAnderson)

****Correct but rejected GAF:**
(foundingDate KarenNationalUnion
(MonthFn April (YearFn 1947)))

The verification step produces comparatively few false negatives (in which a true fact is incorrectly classified as false); in this run, 80% of the *novel, correct* facts retrieved were correctly identified as such. Given this, it is reasonable to reject all unverifiable sentences, especially given the wealth of possible queries and the size and breadth of the corpora available. Only 61% of the *incorrect* facts retrieved were identified, suggesting that substantial work in decreasing the occurrence of false positives will be necessary before the need for human review is eliminated; this is unsurprising, as the Internet contains large amounts of unstructured, unchecked information.

Slightly over a third of the GAFs discovered were facts that were already known to the KB, and presumably correct to a baseline level (*i.e.*, the correctness level achieved by human ontologists); the total number of correct facts discovered was therefore 425, 42% of the total. Verification reduces the number of novel sentences that must undergo human review from 1016 to 61, and the human review process, which takes place entirely in English, is quick and straightforward. An intermediate step towards full automation would be to identify *classes* of sentences that can be asserted without human review.

5 Conclusions

While great strides have been made in machine learning in the last few decades, automatically gathering useful, consistent knowledge in a machine-usable form is still a relatively unexplored research area. The original promise of the Cyc project – to provide a basis of real-world knowledge sufficient to support the sort of learning from language of which humans are capable – has not yet been fulfilled. In that time, information has become enormously more accessible, due in no small part to the widespread popularity of the Web and to effective indexing systems such as Google. Making use of that repository requires a store of real-world knowledge and some facility for natural language parsing and generation.

These results, while extremely preliminary, are encouraging. In particular, using Cyc as a basis for learning is effective, both in guiding the learning process and in representing and using the results. Pre-existing knowledge in the KB supports the construction of meaningful queries and provides a framework into which learned knowledge can be asserted and reasoned over. Comparatively shallow natural language parsing combined with the type constraint and relation knowledge in the Cyc system allows the retrieval, verification, and review of unconstrained facts at a higher rate than that achieved by human knowledge representation experts working unassisted. Perhaps more importantly, the kind of knowledge retrieved is exactly the instance-level knowledge that should not require human experts – it should instead be obtained, maintained, and reasoned over by tools that need and use that knowledge. Involving Google in every stage of the learning process allows us to exploit both Cyc's knowledge and the knowledge on the web in an extremely natural way.

The work being done here is immediately useful as a tool that makes human knowledge entry faster, easier and more effective, but it also provides a basis for analysis of what information can be learned effectively without human interaction. Thus, over time, we hope to provide Cyc with a mechanism to truly acquire knowledge by learning.

Acknowledgments

This research was partially sponsored by ARDA's AQUAINT program. Additionally, we thank Google for allowing access to their API for research such as this.

References

[Belasco *et al.*, 2004] A. Belasco, J. Curtis, RC Kahlert, C. Klein, C. Mayans, R. Reagan. Representing Knowledge Gaps Effectively. In *Proc. of the 5th International Conference on Practical Aspects of Knowledge Management*, Vienna, Austria, p.159-164. Dec 2004.

[Brin and Page, 1998] Sergey Brin and Larry Page, Anatomy of a Large-scale Hypertextual Search Engine. In *Proc. of the 7th International World Wide Web Conference*, pp 107-117, Brisbane, Australia, Apr 1998.

[Brown, 1996] R.D. Brown, Example-Based Machine Translation in the Pangloss System. In *Proc. of the 16th International Conference on Computational Linguistics*, pp 169-174. Copenhagen, Denmark, August 5-9, 1996.

[Charniak, 2001] E. Charniak. A Maximum-Entropy-Inspired Parser. In *Proc. of the 1st conference on North American chapter of the Association for Computational Linguistics*, pp 132-139. Seattle, WA, 2000. Morgan Kaufmann Publishers.

[Etzioni *et al.*, 2004] O. Etzioni, M. Cafarella, D. Downey, A, Popescu, T. Shaked, S. Soderland, D. Weld, A. Yates. Web-scale Information Extraction in KnowItAll. In *Proc. of the 13th international conference on World Wide Web,* pp 100-110, New York, NY, 2004.

[Ghani, 2000] R. Ghani, R. Jones, D. Mladenic, K. Nigam, S. Slattery. Data Mining on Symbolic Knowledge Extracted from the Web. In *Proc. of the 6th International Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, pp 29-36, Boston, MA, Aug 2000.

[Guha, 1991] R.V. Guha. Contexts: A Formalization and Some Applications. PhD thesis, Stanford University, STAN-CS-91-1399-Thesis, 1991.

[Kwok *et al.*, 2001] C. Kwok, O. Etzioni, D. Weld. Scaling Question Answering to the Web. In *ACM Transactions on Information Systems*, Vol 19, Issue 3, pp 242 – 262. 2001

[Lenat, 1976] D.B. Lenat. AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search, Ph.D. Dissertation, Stanford University, STAN-CS-76-570, 1976.

[Lenat *et al.*, 1983] D.B. Lenat, A. Borning, D. McDonald, C. Taylor, S. Weyer. Knoesphere: Building Expert Systems with Encyclopedic Knowledge. In *Proc. of the 8th International Joint Conference on Artificial Intelligence*, Vol 1, pp 167–169, Karlsruhe, Germany, August 1983.

[Lenat, 1995] D.B. Lenat. Cyc: a Large-Scale Investment in Knowledge Infrastructure. In *Communications of the ACM*, Vol 38, Issue 11, pp 33-38. Nov 1995.

[Lenat, 1998] D.B. Lenat, The Dimensions of Context-Space, from http://www.cyc.com/doc/context-space.pdf.

[Panton *et al.*, 2002] K. Panton, P. Miraglia, N. Salay, R.C. Kahlert, D. Baxter, R. Reagan. Knowledge Formation and Dialogue Using the KRAKEN Toolset. In *Proc. of the 18th National Conference on Artificial Intelligence*, pp 900-905, Edmonton, Canada, 2002.

[Prager *et al.*, 2000] J. Prager, E. Brown, A. Coden, D. Radev. Question Answering by Predictive Annotation. In *Proc. of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp 184-191. Athens, Greece, 2000.

[Thrun *et al.*, 1998] S. Thrun, C. Faloutsos, T. Mitchell, L. Wasserman. Automated Learning and Discovery: State-Of-The-Art and Research Topics in a Rapidly Growing Field, tech. report CMU-CALD-98-100, Computer Science Department, Carnegie Mellon University, 1998.

[Witbrock *et al.*, 2003] M. Witbrock, D. Baxter, J. Curtis, D. Schneider R.C. Kahlert, P. Miraglia, P. Wagner, K. Panton, G. Matthews, A. Vizedom. An Interactive Dialogue System for Knowledge Acquisition in Cyc. In *Proc. of the 18th International Joint Conference on Artificial Intelligence,* Acapulco, Mexico, 2003.

[Witbrock *et al.*, 2004] M. Witbrock, K. Panton, S. Reed, D. Schneider, B. Aldag, M. Reimers and S. Bertolo. Automated OWL Annotation Assisted by a Large Knowledge Base. In *Workshop Notes of the 2004 Workshop on Knowledge Markup and Semantic Annotation at the 3rd International Semantic Web Conference,* Hiroshima, Japan, pp 71-80. Nov 2004.

[Witbrock *et al.*, 2005]: M. Witbrock, C. Matuszek, A. Brusseau, R.C. Kahlert, C.B. Fraser, D. Lenat. "Knowledge Begets Knowledge: Steps towards Assisted Knowledge Acquisition in Cyc," in *Proc. of the AAAI 2005 Spring Symposium on Knowledge Collection from Volunteer Contributors*, Stanford, CA, March 2005.