Knowledge Acquisition from Simplified Text

Kevin Livingston and Christopher K. Riesbeck
EECS Dept.
Northwestern University
2133 Sheridan Rd.
Evanston, IL 60208, USA
Tel: 1-847-467-4399 and 1-847-491-7279
E-mail: {livingston; c-riesbeck}@northwestern.edu

ABSTRACT
The problem of entering and integrating new knowledge into a logic-based knowledge base is substantial. Our solution is to provide a natural language interface, which reads simplified English, enabled by a knowledge-based memory-retrieval driven natural language understander. This paper presents a set of tools and interfaces for interacting with such a system, and a discussion of the underlying Reader system, the reading component of the Learning Reader project. The interfaces presented provide direct feedback about what portions of the text are understood, and what interpretations are being produced from it. In addition tools are presented to, among other things, provide example sentences, to facilitate users producing simplified English text suitable for the Reader.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Knowledge Acquisition, Natural Language Understanding, Simplified English, Learning Reader, DMAP

INTRODUCTION
Entering and integrating new knowledge into a logic-based knowledge base is a difficult and arduous task for users. Our solution to this problem is to provide a natural language interface, capable of reading simplified English, enabled by a knowledge-based memory-retrieval driven natural language understander. This paper presents a set of tools and interfaces for interacting with such a system, as well as a discussion of the underlying Reader system, the reading component of the Learning Reader project. The interfaces presented provide direct feedback about what portions of the text are understood, and what interpretations are being produced from it. In addition tools are presented to, among other things, provide example sentences, to facilitate users producing simplified English text suitable for the Reader.

IUI'07, January 28–31, 2007, Honolulu, Hawaii, USA.

The goal of the Learning Reader project is to build a system capable of reading simplified English texts, and integrating the information contained therein into a large existing knowledge base. It is a stated goal of the Learning Reader research to provide leveraged knowledge acquisition by utilizing that which is already known. The Reader component of the Learning Reader is intended to provide a natural form for knowledge entry, not a full-strength language understanding system.

The knowledge acquisition (KA) task is typically arduous and often requires the work of both knowledge engineers and domain experts. Project Halo estimates that encoding a single page of information, scientific content at the AP level, for example AP Chemistry text, costs approximately $10,000. [13] They further estimate that tools to allow for domain experts to more efficiently encode and question the understanding of knowledge systems would bring this cost down significantly, so that it would be more on par with authoring a textbook on the subject.

Since knowledge acquisition through the cooperation of domain experts and knowledge engineers is impractical, research has been done to build systems usable by domain experts alleviating the necessity for a knowledge engineer. SHAKEN [3] is one such system which provides graphical interfaces to the represented knowledge, similar to concept maps. Clark and his colleagues report that this did simplify the KA task, when users were given a week of training before representing new knowledge with SHAKEN.

Cycorp has addressed this problem through form-based knowledge entry tools [17]. These tools leverage knowledge in the Cyc ontology [10] and facts about the structure of various concepts to prompt the user for more information on a topic or make educated guesses about facts for the user to confirm. For example, if the user is introducing Cyc to a new Major League Baseball team, Cyc, knowing that all teams have pitchers will ask who the pitchers are. If the user enters an unknown name, Cyc might then hypothesis that this

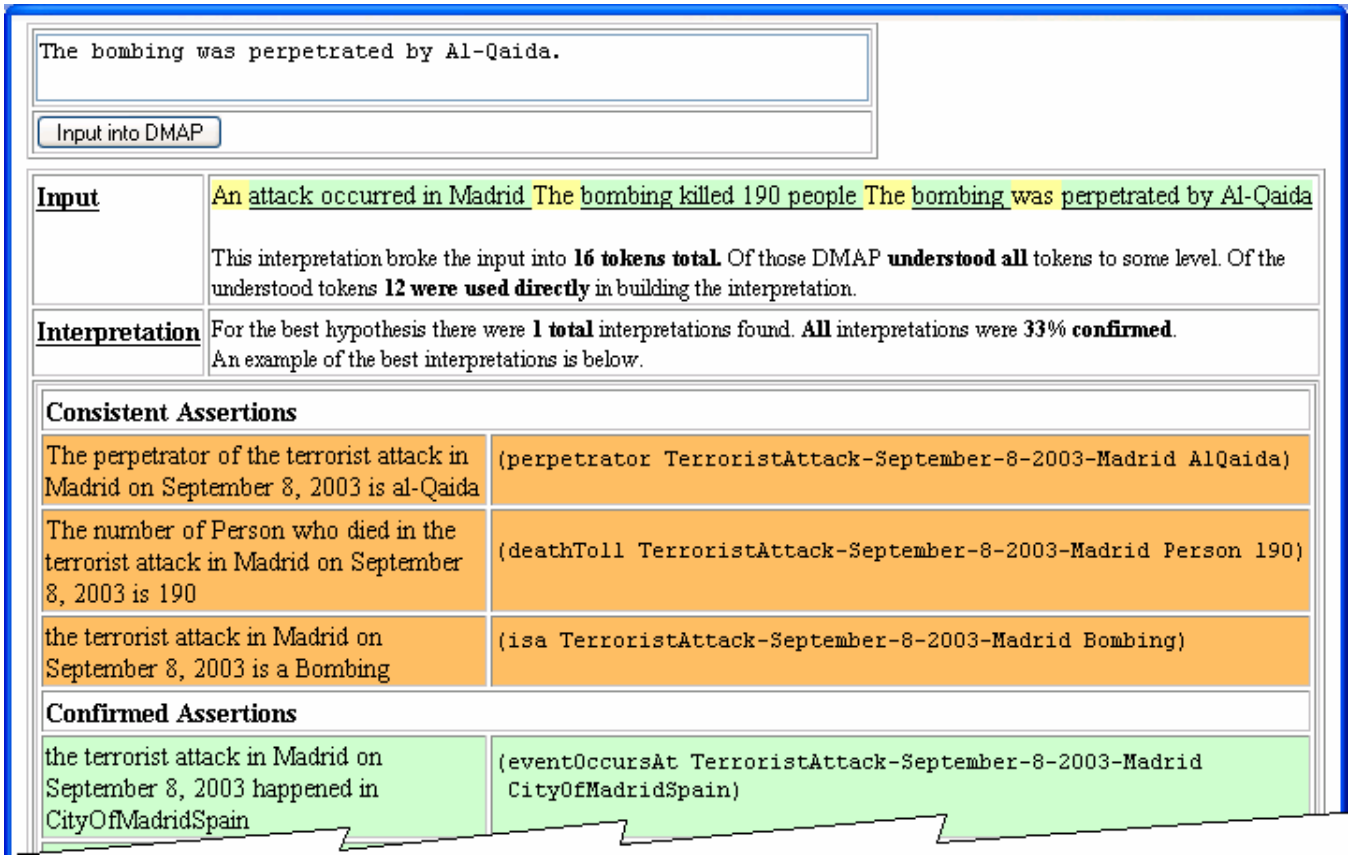

Figure 1: Shows an interface for the Reader, after reading 3 sentences (shown in the "Input" section), as well as a pidgin English interpretation of the .interpretation produced by the Reader (left) and CycL interpretation (right).

new name is a Male-Human knowing that all Major League Baseball players are male, and it would ask the user to confirm this assumption.

Although both SHAKEN and the Cycorp system alleviate the need for both a knowledge engineer and a domain expert, these approach still require placing additional, and significant, burden on the domain experts to either learn new interfaces or vocabularies and ontologies for representing knowledge. It is our goal with the Learning Reader project to present a natural language interface for acquiring such deep knowledge.

The MIT Open Mind Common Sense project takes a completely different approach. [15] Users enter English sentences into Open Mind, either free form or with the assistance of parameterized forms. This text is then mined for information using a "library of handcrafted lexico-semantic pattern-matching rules" [9], arguably the easiest approach of all, as it requires no additional burden on the domain experts to enter knowledge. The pragmatic information mined from Open Mind has been leveraged in several applications [8]. Open Mind is designed to acquire knowledge capable of plausible commonsense inference in context, but not to support a general purpose logical inference engines, such as Cyc [10] or Fire [5].

One interesting alternative to engineering large knowledge bases is to create a kind of virtual KB by using IR techniques on large bodies of text. The work of Hovy et. al. [6], and others in the TREC competition treat the web as a virtual knowledge base and attempt to mine the answers to questions from it. The work of Etzioni et. al. with KnowItAll [4] takes the techniques of TREC one step further by giving the system a set of questions to proactively search for answers, and build a knowledge base of facts. These systems are driven top-down by pre-specified knowledge goals to combine information from many texts, while Learning Reader is driven bottom-up by whatever information is present in a given input text. These systems, like Learning Reader, use language patterns to understand text, however, they typically use a small number of patterns and large corpora of text. Learning Reader on the other hand uses a large number of patterns to operate on small portions of text. The language patterns used by these systems are usually syntactic in nature, for example, expecting a noun phrase, where as Learning Reader patterns are semantic, for example, expecting a reference to an agent or an event.

There has been much research on mapping language onto database access [1], which can produce highly functional systems, such as Yates et. al. interfaces for appliances [16]. However, these approaches requires at least some (if not a lot

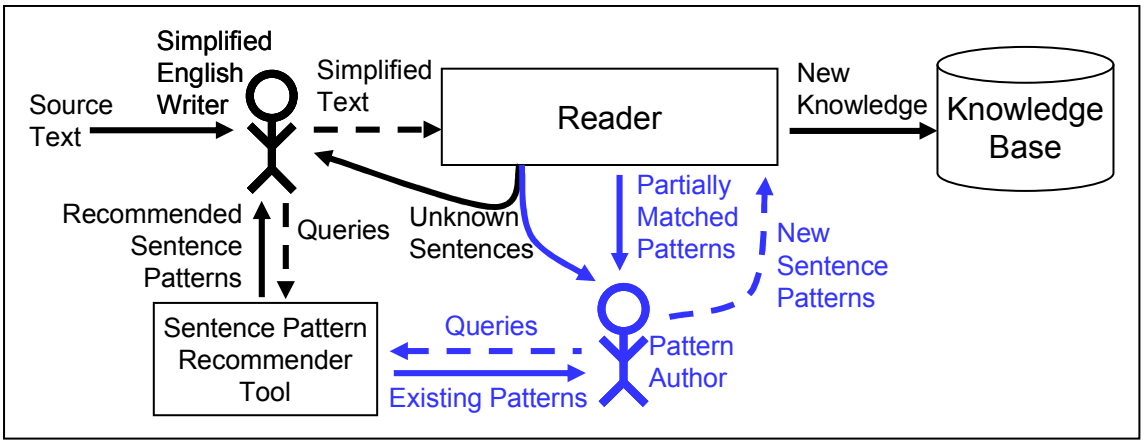

Figure 2: Flow of information between users and Reader. The path across the top depicts the ideal use case for a text simplifier. Blue items indicate interactions with pattern authors.

of) learning and adaptation by users to learn and understand the vocabulary and the grammar necessary to communicate with these systems. Systems of this type typically are designed for a specific context or a restricted domain, although the current primary domain for Learning Reader can be construed as geography, history, and political stories, we make no assumptions which require a restricted domain.

Other techniques which have grown out of the Natural Language Parsing traditions such as Unification Grammars [2] have allowed for the production of systems like CELT [12]. These systems have (a near) one to one mapping between lexical structures and semantic structures produced, which effectively defers the knowledge integration task. By not accepting this constraint and performing knowledge integration at reading time, Learning Reader can leverage its memory to assist in understanding the text.

READER EXAMPLE

An example interaction with the interactive Reader interface is shown in Figure 1. In this example the user has entered the following text.

```
An attack occurred in Madrid.
The bombing killed 190 people.
The bombing was perpetrated by Al-Qaida.
```

The Reader identifies the following assertions as present in or inferable from existing knowledge. Including an existing instance in memory representing the event being referred to, `TerroristAttack-September-8-2003-Madrid`.

```
(isa CityOfMadridSpain (CityInCountryFn Spain))
(isa CityOfMadridSpain CapitalCityOfRegion)
(isa 190 PositiveInteger)
(isa AlQaida TerroristGroup-Islamist)
(isa AlQaida InternationalOrganization)
(isa Person Collection)
(isa Spain Country)
(eventOccursAt
     TerroristAttack-September-8-2003-Madrid
     CityOfMadridSpain)
```

The Reader proposes the following assertions. They represent new knowledge that would not contradict existing knowledge.

```
(perpetrator
       TerroristAttack-September-8-2003-Madrid
       AlQaida)
  (deathToll
       TerroristAttack-September-8-2003-Madrid
       Person 190)
  (isa TerroristAttack-September-8-2003-Madrid
       Bombing)
```

In this case the Reader has extended the existing instance in memory `TerroristAttack-September-8-2003-Madrid` to now be described as a bombing, it now knows that Al-Qaida performed the attack, and how many people were killed.

The system also parrots back its understanding of what it was told in a pidginized version of English. In this manner even users who do not know the underlying ontology and semantics of the memory can view and understand the interpretation being formed by the Reader. In the above example the new information identified by the Reader is presented as follows.

```
The perpetrator of the terrorist attack in Ma-
  drid on September 8, 2003 is al-Qaida.
The number of Person who died in the terrorist
  attack in Madrid on September 8, 2003 is 190.
The terrorist attack in Madrid on September 8,
  2003 is a Bombing.
```

USER INTERACTION

Two different types of users interact with the Reader component of the Learning Reader: text simplifiers and pattern authors. Figure 2 depicts these interactions. The path across the top represents the ideal use case for text simplifiers. They transform source text into simplified text for input into the Reader. The Reader processes this text and produces new knowledge that it integrates and stores in the knowledge base.

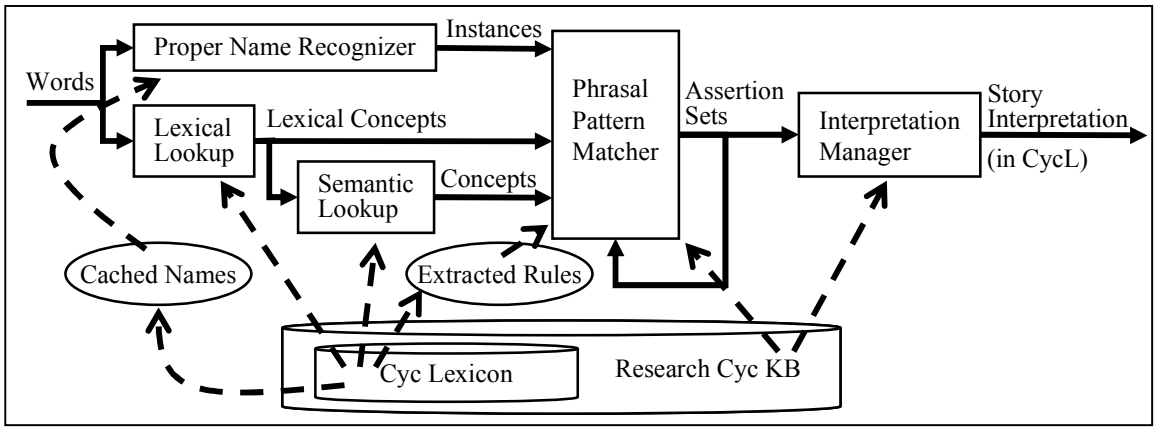

Figure 3: A functional depiction of the DMAP architecture. Boxes represent processes, and solid arrows the flow of data through the system. Dotted arrows represent interactions with the KB or cached portions of it

Sentences that are not understood by the Reader are reported back to the text simplifier, who can then attempt to rewrite these sentences for the Reader. To prevent a trial and error style of interaction, a tool was created to recommend sentence patterns and show example sentences which can be understood by the Reader. The recommender tool (Figures 5 and 6) is discussed in greater detail in the section titled Example Sentence Recommender Tool.

In addition to text simplifiers, pattern authors also interact with the Reader. Pattern authors extend and add patterns for understanding simplified English. Pattern author interactions (depicted in the lower right section of Figure 2, colored blue) are driven primarily by sentences that could not be understood by the Reader. Pattern authors can see which patterns are partially matched by the Reader, and issue queries to the recommender tool to see what other existing patterns are close to the sentence or sentences that the Reader could not understand. In some cases the pattern authors will modify existing patterns, in others they may create entirely new patterns, in order to extend the language covered by the Reader.

IMPLEMENTATION

To be able to use knowledge to acquire new knowledge via language, the Reader is designed to retrieve and apply existing knowledge as early in the process as possible. To do this, the Reader is based on DMAP, a model of language understanding driven by memory structures [11, 14]. DMAP was originally implemented on top of hierarchical frame based memory structures, our implementation is built for logical memories, so that we can interact with Research Cyc [10]. Research Cyc contains over 1.2 million assertions providing DMAP with a large episodic memory to leverage for understanding, and for integrating new knowledge to existing knowledge. As Research Cyc is over two orders of magnitude larger than previous memories used with DMAP systems, issues of scale have arisen. We provide a limited discussion of scale in this work, but more complete discussion of our methods to control combinatorics warrants a paper of its own, more appropriate for other venues.

Figure 3 shows the architecture of DMAP. Textual input is tokenized as a sequence of words. Those words are fed into two processes, a proper name recognizer which would map the string "United States" to the instance `UniedStates-OfAmerica`, and a lexical lookup process which identifies Cyc lexical concepts that correspond to the input text, for example mapping "death" to `Death-TheWord`. Those lexical concepts are then fed to another process which identifies Cyc semantic concepts that correspond to the lexical concepts, for example mapping `Death-TheWord` to the Cyc concept `Deceased`. These processes leverage assertions in the Cyc knowledge base that map strings to corresponding lexical concepts or named entities.

The outputs of all three of these processes are fed into the DMAP phrasal pattern matcher. Functionally the pattern matcher in DMAP is similar to a tabular chart parser [7]. The pattern matcher has mappings from sequences of strings, lexical, and semantic concepts, to semantic assertions which they can be translated into. Figure 4 shows two such mappings. The first sentence is about where an event occurred. The pattern starts by expecting a reference to an `Event`, followed by an expectation for the lexical concept `Occur-TheWord`, etc. The pattern matcher is recursive, and output from one pattern can be used to contribute to a larger pattern. If no patterns are matched DMAP will skip processing that sentence and move on to the next one.

The assertions sets produced by the pattern matcher are fed into an interpretation manager, which is responsible for linking assertions from one sentence to the next, and identifying relevant remindings in the existing memory (KB). Alternate interpretations of ambiguous texts are captured as multiple competing states. DMAP leverages existing memory to help reduce ambiguity by preferring interpretations that reference already existing memory structures.

The critical portion of merging the assertions from one sentence to the next is the alignment of the role variables they contain. This process is combinatorically expensive. DMAP requires that two variables agree with respect to type for them to be merged. Two types agree if one is a

Post Sentence 1 Result
"An attack occurred in Madrid.

Pattern:

```
<Event> Occur-TheWord In-TheWord
<GeographicLocation>
```

Assertions:

```
(eventOccursAt ?event ?location)
(isa ?event AttackOnObject)
(isa ?location GeographicLocation)
```

Variable	Values
?location:	• CityOfMadridSpain
?event:	• TerroristAttack-August-8-2000-Madrid-Spain • TerroristAttack-September-8-2003-Madrid • TerroristAttack-11-mar-2004-Madrid-Spain

Post Sentence 2 Result
"The bombing was perpetrated by the ETA."

Pattern:

```
<Event> Perpetrate-TheWord By-TheWord
<IntelligentAgent>
```

Assertions:

```
(perpetrator ?bombing ?agent)
(isa ?bombing Bombing)
(isa ?agent IntelligentAgent)
```

Variable	Values
?agent:	• BasqueFatherlandAndLiberty
?bombing:	*no values*

Note: the Cyc lexicon defines "ETA" as BasqueFatherlandAndLiberty

Figure 4: Shows the state of DMAP after 2 sentences have been read.

generalization of the other, or if DMAP has, in memory, an example of an instance that is a member of both types; again using its memory to determine what parsing alternatives are acceptable and preferred. DMAP will also prefer those alignments that allow it to retrieve more known facts from memory when the bindings from reading are substituted into the assertions. In lieu of that, DMAP will select an alignment that agrees with respect to types, and does not cause the contradiction of existing knowledge when the bindings are substituted. For example, DMAP considers it a contradiction if an assertion is being formed indicating one agent performed an event, when it has knowledge in memory indicating that another actor performed the event. In the example in Figure 4 there are 7 possible alignments of the variables for DMAP to consider, ultimately it will prefer the following alignment.

```
?location :: <no pair>
   ?event :: ?bombing
<no pair> :: ?agent
```

If DMAP can find no remindings in memory for a given interpretation it will produce new Skolem constants as values for any role variables without values retrieved from memory.

SYSTEM DETAILS

This version of DMAP has over 64,000 proper noun phrases, referring to over 43,000 unique entities, over 36,000 lexical entries, referring to over 19,000 unique lexical concepts, mappings for over 8,000 lexical concepts to over 10,800 semantic concepts, and finally over 28,000 phrasal patterns. These items were all mined from the Research Cyc KB. We built a tool for transforming this information into the phrasal patterns needed by DMAP (examples in Figure 4). This version of DMAP also has 50 handmade patterns, for special forms, such as introducing new named instances, as in, 'There is a city called "Alexandria".' DMAP also makes use of a stemmer so that it can recognize inflected forms of the 36,000 lexical entries.

EVALUATION CORPUS

To evaluate DMAP we have been accumulating a test corpus of simplified English text. Presently the test corpus consists of 62 stories, broken into 183 single topic "snippets". Each snippet contains on average 5 sentences, with the smallest snippet containing 1 sentence and the largest containing 30 sentences. The test corpus contains a total of 956 sentences. The current domain is stories, primarily about world political events, mostly in the Middle East. The text is written using a large vocabulary but (generally) a relatively small number of simple English syntactic structures. Prototypical sentences would be, "Egypt shares a border with the Mediterranean Sea." or "A bombing occurred in Bali.".

PERFORMANCE

The topics of precision and scale of the DMAP algorithms is extensive enough that it warrants its own paper (in preparation for submission to other venues), however, a brief description is provided here for context.

Precision

Correctness is evaluated by comparing the results of reading the 183 snippets for each implementation against a reference set of results. The reference results were produced from an exhaustive (and slow) version of DMAP with a hand tuned set of rules to reduce noise in the output. The results produced were inspected manually to remove any assertions deemed erroneous. These reference results do not yet represent a complete encoding of the corpus in CycL (the representation language of Cyc), nor is every sentence in the corpus even necessarily completely representable in the currently available Cyc ontology. For example, one of the sentences in the corpus, "Other littoral states have generally agreed to equidistant seabed boundaries," would challenge even skilled human CycL authors.

Of the 62 stories in the corpus, broken into 183 single topic snippets, containing a total of 956 sentences, our best algorithm produces assertions for 671 sentences (70%), failing to interpret 258 sentences, spread over 51 stories, and 119

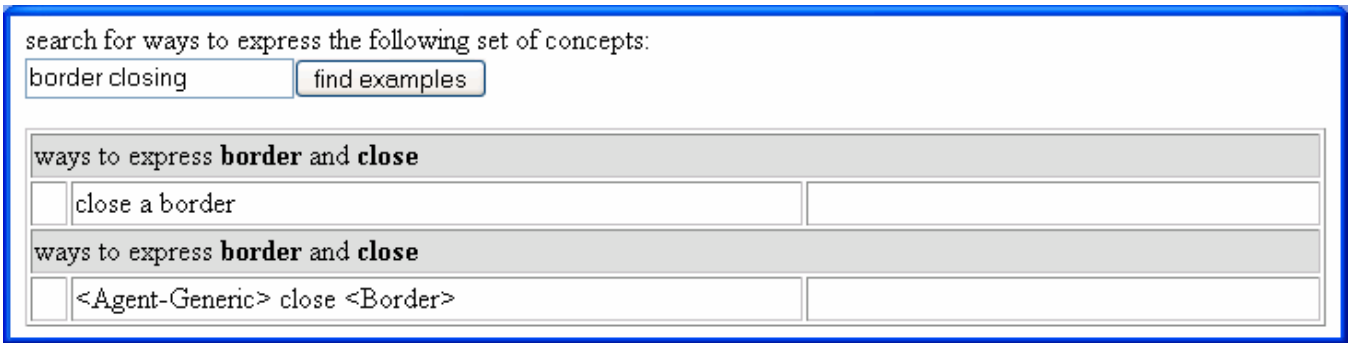

Figure 5: Shows pattern recommendations for the query "border closing"

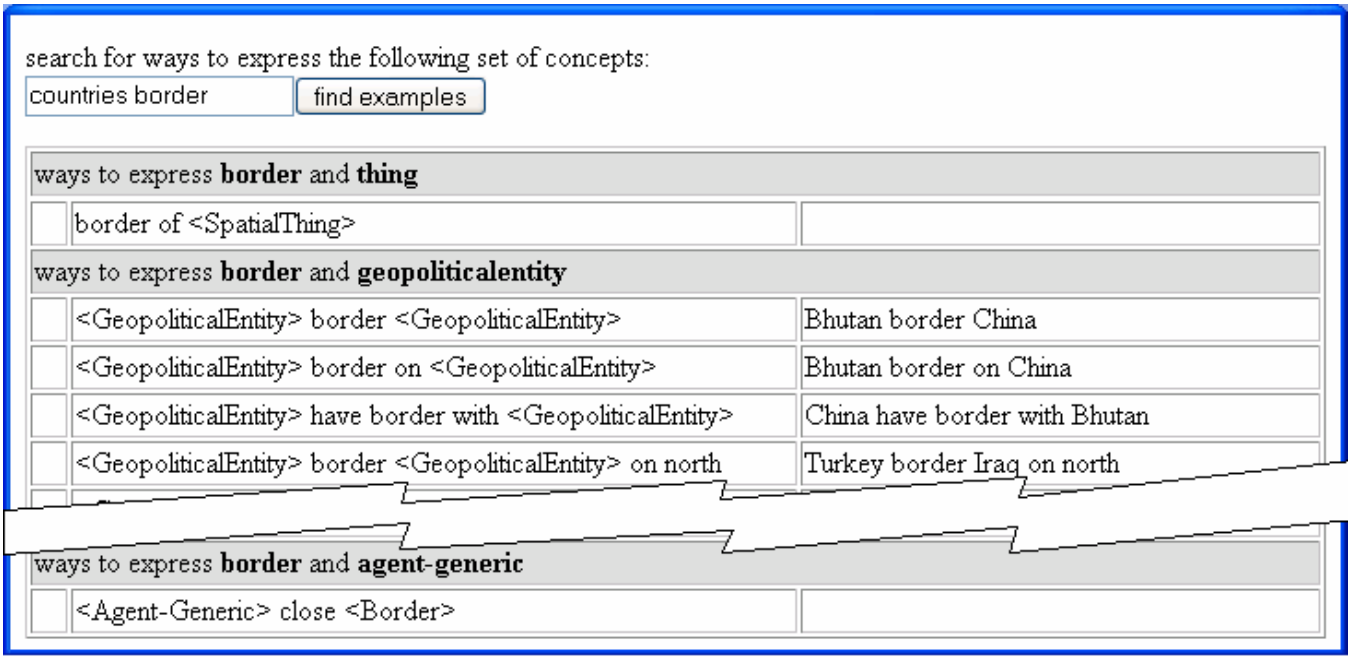

Figure 6: Shows pattern recommendations for the query "countries border" with general patterns on the left and example pidgin English sentences on the right.

snippets. Failures could be caused by no patterns matching the input sentence, or by processing limits being reached, and DMAP being forced to move on to the next sentence. This algorithm correctly reproduced 84% of the assertions produced by the reference algorithm. Improving performance is the subject on ongoing research.

Speed and Scalability

Since DMAP in the Learning Reader project is intended to function effectively in real time, and since many of the steps in the underlying algorithms are theoretically exponential, controlling scale and running time are critically important.

The same algorithm mentioned previously was able to process 93% of the corpus in under 4 seconds per sentence (77.4% of sentences in under 1 second each). The remaining 7% took over 4 seconds with the maximum being a little over 2 minutes.

Memory(KB) Coverage

In this version with the current set of NL information extracted from Cyc, DMAP can find assertions in memory through textual reference, involving 99% of the collections in the Cyc ontology. DMAP can produce new assertions involving 57% of the collections.

With respect to predicates in the Cyc ontology, DMAP has phrasal patterns that can produce and access 13% of all predicates. However, if we include only predicates that have instances of assertions in memory, coverage increase to 15.4%.

Looking at predicate coverage more closely, if we break this down by collection, and put all the assertions made about the instances of a collection into separate bins. We get 6,704 sets of assertions. Of those 4,480 sets have predicates which are covered by DMAP the mean coverage is 41% and the median is 39%. The distribution is linear across those collections, with 184 having 100% of their predicates covered, and 1532 having above 50% covered.

This indicates that even though overall predicate coverage is low, there is a wide distribution of NL knowledge in Cyc, with some areas being described well, and others not so well.

In terms of assertions, DMAP can access 43% of the 1.2 million assertions presently made in Research Cyc. This number includes isa expressions. If we only consider predicates other than isa, DMAP can access 24.8% of Cyc.

ASSISTANCE IN AUTHORING SIMPLIFIED ENGLISH

We have presented the structure and an implementation of a natural language understanding system, which can acquire new knowledge, in deep and structured semantics, integrated with existing episodic knowledge in a large existing knowledge base (Research Cyc). The Reader allows for open domain, open vocabulary, understanding of simplified English text. The Reader currently acquires knowledge primarily about instances, for example, descriptions of current events, or the knowledge about the borders between two countries. It does not yet handle the learning of general knowledge or rules such as "all armies have weapons". Representing such assertions requires more complicated concepts to be represented, such as quantification. This is a future goal of the Learning Reader research.

Since some concepts and topics are covered more sparsely than others (see Memory Coverage section), it becomes important to have tools to efficiently direct those writing simplified English for the Learning Reader to the correct phrasal patterns for what they are trying to express. The interaction with these tools is done primarily in English and does not require knowledge of the underling ontology or memory, although the tools provide even more features for those who are familiar.

In addition to tools to assist in writing simplified English, tools have also been created to assist developers in evaluating coverage of the knowledge base. These tools such as those used to perform the analysis in the Memory Coverage section, can be used to illuminate coverage gaps in either the Reader algorithms or the phrasal patterns which drive them.

Reader

The Reader interface (see Figure 1), allows users to enter sentences one at a time, and view the Reader's interpretation of the story after each sentence is processed. The Reader presents its understanding both in the form of pidgin English sentences, and CycL assertions.

The Reader displays the text it has been given annotated to indicate what it understood and used. Text that was understood and that contributed to the final interpretation is underlined and colored green. Words that are in the Reader's vocabulary, but not used in pattern matching are highlighted yellow. Words which are completely unknown (and obviously unused) are drawn in italics and highlighted red. Referring to the example in Figure 1 it can be seen that DMAP understood all the words in the example to some level, but did not use the words "An", "The", or "was" in the pattern matching step.

The current system does not learn vocabulary online, except for new names, which are introduced by sentences in the form of "There is an X called Y". However, it is a subject of future research to be able to learn and understand new vocabulary.

Example Sentence Recommender Tool

In addition to the primary Reader interface, a set of tools have been developed to assist users in the generation of simplified English appropriate for the Reader. An example of one such tool can be seen in Figures 5 and 6.

The Reader system is built on top a large set of phrasal patterns (over 28,000) extracted from Cyc, however, those patterns still only represent a fraction of the syntactic and semantic variations in English. The Reader interface (Figure 1) attempts to make it easy to see when sentences are not understood, indicating when a sentence needs to be rewritten. Although, when a parsing failure is encountered it sometimes can be difficult for the user to know how to rewrite a sentence so that the Reader will be able to capture its meaning. The tool showing in Figures 5 and 6 is a way for users to ask, "how do I express X?".

This tool allows users to enter a sequence of concepts in English. These terms are then parsed and translated through the same front end parsing techniques that feed DMAP. This produces a set of related concepts and predicates in the Cyc ontology. However, instead of feeding these concepts to the pattern matcher, the pattern matching rules, used by DMAP, are searched to identify those that either use the given words in their pattern, or refer to the denoted semantics in their output. Inference is also done to search the specializations and generalizations of the concepts identified from the query text.

Two examples relating to "borders" are shown. In the shorter example, the user queried for "border closing". The system responded with two examples, one for the phrase "close a border" and one about an agent or actor closing a border. In the longer example, the user queried for "countries border". In this case the interface presents a set of examples relating primarily to how express two geographic regions or geopolitical agents bordering each other. The tool recognizes these patterns as relevant, because it knows that countries are types of both geographic regions and geopolitical agents. On the right side of the interface, in many instances, the tool was able to identify instances from memory that relate to the retrieved pattern, and so it automatically constructed pidgin English "sentences" of those examples, which are a minimal representation of what is required for that pattern to be matched by DMAP. Note the pattern about an agent closing a border also appears in this set too. This is because the tool understands that countries could operate as agents (at least in the Cyc ontology), and therefore it concluded that this pattern was also relevant.

After identifying relevant phrasal patterns, the user can go back to the Reader, input a new sentence, and verify that it was understood correctly. Tightening this loop is an obvious subject of future research.

FUTURE WORK

A major focus will be to extend the tools for writers generating simplified English to provide even more support for users. One obvious extension is to couple the Reader and the tool that recommends example phrases and sentences. This would provide two benefits. First the sentence recommender could leverage the context of what the Reader has read so far to produce better recommendations. Second, when the Reader fails to interpret a sentence, it can fail soft, by issuing a query to recommender based on the words in the sentence it was trying to read. The results could be presented to the user to help in the rewriting of the sentence they were trying to convey to the Reader, possibly even doing much of the "typing" for the user.

In addition work will be done to improve the algorithms of DMAP to extending coverage. Also, a new version of Research Cyc is also becoming available with nearly twice as many assertions, providing more episodic memory to leverage during understanding time. Brief inspection also shows that this version of Cyc has more natural language knowledge as well, which hopefully can be put to good use by DMAP.

SUMMARY

This paper presents the Reader, a natural language interface for knowledge acquisition. The Reader operates on simplified English text, and we present a suite of tools for assisting users in the production of simplified text for processing by the Reader.

An interface for the Reader was presented which allows the user to see what text was and was not understood, and how it is interpreting what is being read. Additional tools were also presented which help users to identify natural language examples for expressing information they wish to convey to the Reader.

We have also discussed how through using DMAP the Reader is able to leverage existing knowledge to assist in the acquisition of new knowledge via language.

ACKNOWLEDGMENTS

The authors would like to acknowledge the other members of the Learning Reader research group for their discussion and help which contributed to this paper: Ken Forbus, Larry Birnbaum, Abhishek Sharma, Leo Ureel, and Dave Raffensperger This project was funded through DARPA, grant HR0011-04-1-0051. We would like to thank Cycorp for access to and support with using Research Cyc. The Qualitative Reasoning Group (QRG) at Northwestern has also been essential to this work through ongoing support with using their Fire reasoning engine.

REFERENCES

1. Androutsopoulos, I.; Ritchie, G.D.; and Thanish, P. Natural Language Interfaces to Databases - An Introduction. *Natural Language Engineering*, vol 1, part 1, 29--81, 1995.

2. Calder, J., Klein, E., and Zeevat, H. 1988. Unification Categorial Grammar: a concise, extendable grammar for natural language processing. *Proceedings of the 12th Conference on Computational Linguistics* - Volume 1 (Budapest, Hungry, August 22 - 27, 1988). D. Vargha, Ed. International Conference On Computational Linguistics. Association for Computational Linguistics, Morristown, NJ, 83-86.

3. Clark, P., Thompson, J., Barker, K., Porter, B., Chaudhri, V., Rodriguez, A., Thoméré, J., Mishra, S., Gil, Y., Hayes, P., and Reichherzer, T. 2001. Knowledge entry as the graphical assembly of components. In *Proceedings of the 1st international Conference on Knowledge Capture* Victoria, British Columbia, Canada, October 22 - 23, 2001.

4. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, 2004 A.: Web Scale Information Extraction in KnowItAll (Preliminary Results*). In: Proceedings of the 13th International World Wide Web Conference*, New York, USA 2004.

5. Forbus, Kenneth D., de Kleer, Johan. *Building problem solvers*, MIT Press, Cambridge, MA, 1993.

6. Hovy, E., Gerber, L., Hermjakob, U., Junk, M., and Lin, C., 2001, Question Answering in Webclopedia, *Proceedings of the TREC-9 Conference.*

7. Kay, M. Chart generation. *In Proceedings of the 34th Annual Meeting on Association For Computational Linguistics* (Santa Cruz, California, June 24 - 27, 1996). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 200-204.

8. Lieberman, H., Liu, H., Singh, P., and Barry, B. Beating common sense into interactive applications. *AI Magazine*, Winter 2004, 25(4):63-76. AAAI Press.

9. Liu, H.; and Singh, P.. Commonsense Reasoning in and over Natural Language. *Eighth International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES2004)*, Wellington, New Zealand, 22–24 September, 2004.

10. Lenat, D. B. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38 (11): 33–38. 1995.

11. Martin, C. E. *Direct Memory Access Parsing.* PhD thesis, Yale University. 1990.

12. Pease, A.; Murray, W., 2003 An English to logic translator for ontology-based knowledge representation languages, *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering, 2003*, vol., no.pp. 777- 783, 26-29 Oct. 2003.

13. Project Halo. http://www.projecthalo.com/

14. Riesbeck, C. K. From Conceptual Analyzer to Direct Memory Access Parsing: An Overview. *Advances in Cognitive Science.* pages 236-258 N. E. Sharkey (ed.), Prentice Hall. 1986.

15. Singh P, Lin T, Mueller E T, Lim G, Perkins T and Zhu W L: Open mind commonsense: knowledge acquisition from the general public, *Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems, Lecture Notes in Computer Science* No 2519 Heidelberg, Springer 2002.

16. Yates, A., Etzioni, O., and Weld, D. 2003. A reliable natural language interface to household appliances. *Proceedings of the 8th international Conference on intelligent User interfaces IUI2003*. (Miami, Florida, USA, January 12 - 15, 2003). ACM Press, New York, NY, 189-196.

17. Witbrock, Baxter, Curtis, Schneider, Kahlert, Miraglia, Wagner, Panton, Matthews, Vizedom. An Interactive Dia-logue System for Knowledge Acquisition in Cyc. *In Proc. of the 18th International Joint Conference on Arti-ficial Intelligence*, Acapulco, Mexico, 2003.