Methods of Rule Acquisition in the TextLearner System

Jon Curtis, David Baxter, Peter Wagner, John Cabral, Dave Schneider, Michael Witbrock

Cycorp, Inc.
{jonc|baxter|daves|witbrock}@cyc.com,
{peterawagner|john.cabral}@gmail.com

Abstract

This paper describes the TextLearner prototype, a knowledge-acquisition program that represents the culmination of the DARPA-IPTO-sponsored Reading Learning Comprehension seedling program, an effort to determine the feasibility of autonomous knowledge acquisition through the analysis of text. Built atop the Cyc Knowledge Base and implemented almost entirely in the formal representation language of CycL, TextLearner is an anomaly in the way of Natural Language Understanding programs. The system operates by generating an information-rich model of its target document, and uses that model to explore learning opportunities. In particular, TextLearner generates and evaluates hypotheses, not only about the content of the target document, but about how to interpret unfamiliar natural language constructions. This paper focuses on this second capability and describes four algorithms TextLearner uses to acquire rules for interpreting text.

Introduction

The Reading Learning Comprehension seedling sought to establish the feasibility of automated knowledge acquisition from and about natural language. This effort produced the TextLearner prototype, which represents a model of learning quite different from most research on learning performed over the past two decades, such as function approximation, reinforcement learning, and Markov model learning, which has focused on statistics-based learning of sequences and patterns.

The approach taken to textual analysis inside TextLearner approximates important features of reading. Among these is the ability to defer resolution of ambiguous words and phrases, and thus "move through" a text, using later information to resolve earlier uncertainties. Towards this end, TextLearner generates an information-rich model of the target text, including a declarative, formal representation of tokenizations, linkages, and parse trees, and stores this representation in a knowledge base primed with thousands of commonsense concepts and rules. This environment—a reified model of context against a backdrop of common sense—enables TextLearner to generate multiple competing hypotheses about the meanings of words, phrases, and sentences, and to reason about their respective merits.

This level of context-modeling also enables TextLearner to approximate another important, if less recognized, feature of reading: the use of context to learn rules that facilitate and improve the process of learning from text. Though it is evident that humans are able to improve both their reading comprehension and writing skills by reading, a precise model of how humans manage this feat was not required to make advances in this area. Rather, familiarity with the core natural language understanding (NLU) capabilities of existing software allowed us to identify small classes of rules that were both tractable to learn and capable of yielding results. This paper[1] describes four such classes of rules and the algorithms implemented in TextLearner for their acquisition. We begin with a brief description of the Cyc system, upon which TextLearner is built; then describe the document modeling that supports rule acquisition.

The Cyc Knowledge Base

TextLearner is built atop the Cyc Knowledge Base, a large repository of common sense and specialized knowledge.[2] Knowledge in Cyc is represented in CycL, a higher-order logical language based on predicate calculus. Every CycL assertion—and hence every sentence provable in the Cyc system—occurs in a context, or *microtheory*, allowing for the representation of competing theories, hypotheses, and fictional claims. Cyc's inference engine combines general

[1] The content extraction capabilities of TextLearner have been described in (Curtis, Baxter and Cabral 2006).
[2] At the time when this article was written, the fully Cyc KB, which is not (yet) publicly available, has over 5 million assertions. A freely available version of Cyc can be downloaded at http://www.opencyc.org/.

theorem proving (rule chaining) with specialized reasoning modules to handle commonly encountered inference tasks.

The higher order features of CycL support both high expressivity and efficient inferencing (Ramachandran, Reagan and Goolsbey 2005). In particular, Cyc is able to reason about the properties of collections, relations, CycL terms (including sentences and assertions), and even the contexts in which reasoning and representation takes place. This level of expressiveness—unobtainable in a first order or purely taxonomic representational scheme—underlies TextLearner's ability to generate the very rules it uses to interpret text.

Document Modeling

A number of distinctions in the Cyc ontology are relevant to TextLearner's approach to modeling its target documents. The most intuitive of these is a distinction between an *authored work*, such as a musical score, a screenplay, or an op-ed piece, and its spatio-temporal *instantiations*—musical broadcasts, theatrical performances, newspaper clippings and the like—by which humans experience the contents of the work. English contains many words that can be used to refer to either a work or a work-instantiation, such as "book," "document," "paper," "article," and "movie." However, Cyc captures the distinction unambiguously via distinct terms that correspond to the two kinds of senses. The CycL term `PropositionalConceptualWork` (PCW) denotes the class of abstract works that convey propositional (truth-evaluable) content, and so has specializations, such as `Book-CW`, that are useful in representing that the "work" sense of "book" is at play in the sentence "I read every book that Stephen Hawking has ever written." In contrast, `InformationBearingThing` (IBT) represents the collection of all objects and events that carry information. Under this collection one finds terms such as `BookCopy`, useful for representing and understanding "instantiation" uses of "book," as in "I lost Stephen's book in the scuffle."

Perhaps the most crucial distinction on which TextLearner relies is that between a PCW and its *information structure*, is represented in Cyc and is central to TextLearner's modeling. The CycL term `AbstractInformationStructure` (AIS) denotes the class of abstract structures that encode information. There is a functional relationship between a PCW at any given point in time and its structure, represented in CycL by the predicate `correspondingAIS`. To arrive at any sort of representation of a document's content, TextLearner must analyze the *structure* of the document. To this end, it reifies an information structure functionally, using the functor `AISForFn` such that for a PCW document 151,

```
(correspondingAIS Doc-151
  (AISForFn Doc-151))
```

`(AISForFn Doc-151)` denotes the actual abstract information structure of document 151—i.e., the very structure shared by all physical printings that all its physical copies must instantiate.

At the most general level of description, the TextLearner procedure for analyzing text involves identifying a PCW and reasoning about its `correspondingAIS`, which, for which, for textual works, is typically a string of characters. In short, if a document D has as its structure a string S, one fully can understand "the propositional content" of D by successfully interpreting S. Though this is a fairly innocuous view to hold at the level of whole documents, it is misleadingly dangerous when extended to smaller parts of documents, such as individual sentences, phrases or words, the meanings of which depend on the surrounding context supplied by the document. Thus to talk of "the meaning" or "the right interpretation" of a word-, phrase-, or even sentence-level *string* is misleading: A given string (e.g., "he") does not carry with it an inherent *right* meaning; rather it is a *contextualized* structure—an *occurrence* of a string within a larger structure—that can be said to have a meaning.[3]

The concept of a contextualized information structure was, prior to the Reading Learning Comprehension project, not explicitly represented in the Cyc ontology. As a result, the term `ContextualizedInformationStructure` (CIS) was added to the Cyc KB. It is possible for there to be many sub-classes of contextualized information structure; however, TextLearner is only concerned with contextualized words, phrases, and sentences. To represent this sub-class, the term `LinguisticExpressionPeg`[4] (LEP) was added to Cyc.

Once TextLearner has the structure of a document fully represented, it begins processing the first paragraph, sentence by sentence, in the order in which the sentences appear. Almost all of the work TextLearner does is at the level of individual sentences; however, the model for each

[3] Lest there be confusion on this point, the abstract information structure/contextualized information structure distinction and the type/token distinction, though analogous, are not identical. In this very footnote, "x" and "x" are two distinct tokens of the same type, and their relative spatial location, combined with the fact that they instantiate the same abstract information structure (the same string) is sufficient to conclude that there are two distinct contextualized information structures (two distinct occurrences of the same string) within the structure of this page. However, that does not mean that the tokens are the same thing as the contextualized structures. If one makes a photocopy of this page, one will have succeeded in generating two more distinct tokens of the same type. But just as all *four tokens* instantiate a *single* abstract information structure (the same string), *each pair* of tokens merely serves as a physical reminder that there are *two* contextualized information structures (two occurrences of that single string) within the single shared structure that both copies of this footnote instantiate. That there is a second pair of tokens, however, does not suggest that there is a second pair of contextualized information structures.

[4] The use of the word "peg" here reflects a similar (but not identical) usage in the works of Bonnie Webber in (Webber 1978), and others, who use "pegs" as objects from which to hang ever-increasing amounts of data as a discourse model is developed.

sentence is maintained after its completion, so that the analysis of each new sentence can benefit from what was learned during the analysis of earlier sentences (and, conversely, so that the system can reassess some of its judgments about the content of earlier sentences in light of what is discovered about later sentences).

The first step in the sentence-analysis process is to reify a sentence-level LEP representing the contextualized occurrence of the sentence-string in the target paragraph of the selected document. In the current implementation of TextLearner, sentences are introduced through "create" operations that generate a CycL constant for the sentence-level LEP. At this point, some of its defining properties are declared in the knowledge base as well. An example sentence-level peg, at this early stage, might thus have this sort of relatively bare representation, identifying it as a structure of the relevant type, and giving its relative location within `Paragraph02`, and its original html source string:

```
Constant: L2R-SentencePeg-1.

isa:
LinguisticExpressionPegCompleteUtterance
TextStringOccurrence.
  nthOccurrenceOfStructureTypeInStructure:
(1 NLSentence Paragraph02).

asHtmlSourceString:
"Ghandi followed a <a ref="/wiki/Vegetarian"
title=Vegetarian"vegetarian</a> diet.".
```

With the reification of a sentence-level peg, TextLearner is ready to subject the sentence to syntactic analysis. To this end, TextLearner calls the Link Parser (Temperely, Sleator and Lafferty 2004) a highly lexicalized, statistically ranked parser for English that produces 1) a tokenization of the input sentence, 2) a linkage, a graphical structure the connects tokens with syntactically and, to some degree, semantically significant links between individual tokens in the tokenization; and 3) a parse tree, the structure of which follows directly from the linkage.

TextLearner reifies all three outputs, introducing new CycL terms for the tokenization, individual tokens, the parse tree, and its nodes. Each token is associated with its string, its corresponding node in the tree, and to other tokens as they are in the linkage, via CycL predicates corresponding to links.

One reason for choosing the Link parser for use by TextLearner is that some links encode a semantic relationship, making them strictly more informative than a standard parse tree, and informative in an especially helpful way, as the semantic meanings of links can be encoded as rules for interpreting text. As will be shown in the section on semantic translation templates, below, some of this semantic information proved useful in acquiring certain rules for semantically translating new verb phrases into CycL.

Rule Acquisition: "Learning to Read"

Though content extraction might be the most recognized and natural objective for systems that implement NLU techniques, an early objective of TextLearner has been that it learn "increasingly autonomously" from text (Witbrock and Schneider 2004). This means that not only should TextLearner perform some level of content extraction, it should learn in a way that makes it do a better job of learning from text over time. Specifically, this entails finding ways to learn rules that help the system handle new linguistic phenomena.

A handful of high-payoff areas for rule acquisition were identified:

- **Rules to Fill Lexical Gaps**. The Cyc Knowledge Base has an English lexicon[5] with substantial, but not complete, coverage. Insofar as sentence-level understanding in TextLearner depends on the ability to understand the constituent words, gaps in the Cyc Lexicon pose an obstacle to sentence-level understanding. Thus TextLearner must be able to hypothesize rules that will fill classes of gaps in response to encountering, at a minimum, parts of speech for words that are not accounted for in the Cyc lexicon.
- **Rules for interpreting noun compounds.** Having generated hypotheses about what the head and modifier of an identified noun compound could mean, the system should generate rules explaining the compound—i.e., rules that extend the semantic relationship between the head and modifier semantics (e.g., the semantics of "rain" and "coat") to explain the meaning of the compound in context (e.g., "rain coat" denotes a type of coat that guards its wearer against rain).
- **Rules for interpreting new verb phrases.** Current mechanisms for translating verb phrases into CycL rely on semantic translation templates. Coverage in this area is substantially incomplete. The system should, when enough information is given in the context model, propose new templates that enable some level of formalization.
- **Rules (templates) to enable Example-Based Machine Translation (EBMT).** EBMT is the process of deriving translation based on examples (Nagao 1984). Though this technology has been used primarily to translate between natural language sentences, some promise has been shown in the area of applying these techniques to effect NL-to-CycL translations (Bertolo 2000), despite obvious issues with respect to the use of natural language devices such as metaphor and pronouns that

[5] For more detail on the Cyc Lexicon, see (Burns and Davis 1999).

do not (typically) correspond to elements of a formal representation language.

The details behind each rule acquisition method are treated in turn.

Lexical Gap-Filling Rules

In cases where the Cyc lexicon contains denotations for some, but not all, parts of speech for a word, TextLearner can draw upon the semantic space surrounding a denotation to find candidates to fill the gap. For example, `Beer-TheWord` is known to have both a count noun form ("beer" and "beers") and a mass noun form ("beer"), but the Cyc lexicon only contains a single denotation handling only the mass noun case.

```
(denotation Beer-TheWord MassNoun 0 Beer)
```

This leaves a gap in the lexicon with respect to any count noun denotation for `Beer-TheWord`. In the context of TextLearner, this means that when considering an occurrence of "beer" that has been identified with a lexical parse tree node marked as having mass number, the lexicon will be unable to provide a possible semantic interpretation. In an attempt to fill this gap, TextLearner examines the relationship of `Beer` to other concepts in the Cyc ontology, not only to fill this particular gap, but also to hypothesize rules that can be used to fill similar gaps of that type (i.e., count noun gaps when a mass noun denotation is known).

For example, Cyc knows the following salient[6] fact about beer:

```
(servingTypeOfFoodOrDrinkType (FoodServingFn
                    Beer) Beer).
```

This sentence identifies `(FoodServingFn Beer)` as the collection of individual beer servings. TextLearner then forms a general hypothesis to explain how the count noun `denotation` gap of `Beer-TheWord` might be filled: The relation `servingTypeOfFoodOrDrinkType` is hypothesized to be a lexical-extension relation, such that when a concept denoted by a mass noun appears in the second argument, the other argument will contain a concept denoted by a count noun version of that noun. This rule will forward-derive a new denotation for `Beer-TheWord`:

```
(denotation Beer-TheWord CountNoun 0
        (FoodServingFn Beer))
```

[6] The notion of salience is notoriously context-sensitive. Though not formalized here, the salience of a relation here is grounded in a handful of heuristics that have proved useful in past Cyc-based applications, notably (Curtis, Matthews and Baxter 2005). In this particular example, the heuristic used might be summarized as follows: *When a functional relationship exists between two distinct classes, and both classes and their relationship has been explicitly represented in the Cyc KB, then that relationship is salient with respect to either class.*

as well as denotations for any other mass noun denotations for which Cyc knows there to be a `servingTypeOfFoodOrDrinkType` relationship, such as "soda," "pizza," "water," and the like. These rules are placed in a special microtheory for later review by trained ontologists, but the `denotation` entries they generate are made immediately available for TextLearner to use.

Noun Compound Rules

TextLearner's first step in interpreting a noun compound involves recognizing cases where a parse-tree includes two adjacent nouns, the first of these having a non-plural form (e.g. "noun compounds" or "Bantu word"). On the basis of such configuration, the system concludes that the CycL predicate `candidateNounCompound` holds between the first node—we'll call this the noun compound modifier—and the second node, the noun compound head. This `candidateNounCompound` assertion, in turn, serves as a (partial) basis for concluding a variety of other assertions that are used to drive both noun-compound interpretations and the hypothesizing of new noun compound rules.

When the system recognizes that the `candidateNounCompound` relation holds between nodes in a tree, it determines whether the syntactic and semantic features of these nodes are compatible with the syntactic and semantic constraints of noun compound rules familiar to the system. On the basis of this determination, some number of `nounCompoundWithRule` assertions will be generated via forward inference. The quintary (arity 5) predicate `nounCompoundWithRule` relates the modifier and head nodes, along with their relevant CycL interpretations, to whatever rules they happen to satisfy.

For a noun compound like "faculty investigation," this process includes determining whether any potential interpretations of "faculty" and "investigation" are consistent with any known rules. In this case, we find that when the modifier is interpreted as `AcademicDepartment` and the head as `Investigation`, these nodes satisfy the constraints of `CourtRuling-NCR`, a rule that can be used to relate specializations of `IntelligentAgent` to specializations of `PurposefulAction` generally.

The above-mentioned `candidateNounCompound` assertions also trigger attempts to derive new (hypothesized) noun compound rules on the basis other (generally non-linguistic) knowledge in the KB. Currently, TextLearner's strategies for doing this capitalize on knowledge that has been encoded using generalizations such as "every word is a word in some language," or, in CycL:

```
(relationAllExists wordInLanguage
   LexicalWord NaturalLanguage)
```

Using such assertions, TextLearner is able to hypothesize novel noun compound rules. One such rule it hypothesized while processing the Wikipedia article, "Angola" was prompted by the unfamiliar phrase "Bantu Word." The resultant rule was formalized as:

```
(NCRuleWithTemplateFn
 (TheList
   SubcollectionOfWithRelationToFn)
 (TheList TheNCHead genls LexicalWord)
 wordInLanguage
 (TheList TheNCModifier isa
   NaturalLanguage)))
```

This rule specifies a recipe for generating a semantic interpretation for noun compounds that meet certain criteria. First, the modifying noun must denote a Natural Language (e.g. "Bantu") and the head must be a noun that denotes a type of word (e.g. "word," "verb," or "name"). It further predicts that a compound which has this form will denote terms of the type specified by the head which are also words of the language specified by the modifier. That is, when TextLearner encounters a noun compound "Bantu word," it will use this rule to interpret it as possibly meaning the collection of words in the Bantu language, or

```
(SubcollectionOfWithRelationToFn LexicalWord
      wordInLanguage BantuLanguageSet)
```

Semantic Translation Templates

Semantic translation (sem-trans) templates are CycL sentences that give a recipe for translating natural language phrases into CycL (Burns and Davis 1999). These templates are used by proprietary cyclification (NL-to-CycL) parsers. The performance of these parsers are thus to a large degree limited by issues of coverage: A particular semantic translation template needed to handle a given use of a verb might be missing, leading the parsers to produce partial or no results. Given that TextLearner calls upon the PSP and Cyclifier to generate candidate semantic interpretations, its ability to extract information from text will also be limited by coverage, making the acquisition of new templates part of its "learning reading" mandate.

As part of the construction of the TextLearner program, some progress has been made in the area of acquiring new verb semantic translation templates in response to encountering prepositional complements. Consider, for example, the sentence

"The heart pumps blood to the lungs."

The main verb of this sentence, "pumps," has an entry in the Cyc Lexicon under `Pump-TheWord` for which the following semantic translation template is known:

```
Mt: GeneralEnglishMt
(verbSemTrans Pump-TheWord 1
TransitiveNPFrame
(and
(isa :ACTION PumpingFluid)
(providerOfMotiveForce :ACTION :SUBJECT)
(primaryObjectMoving :ACTION :OBJECT)))
```

This use of `Pump-TheWord` is transitive ("The heart pumps blood ...") and so there is enough information in the `TransitiveNPFrame` translation template to cover the subject-verb-object relationship; however, there is insufficient information to cover the prepositional complement. In other words, the most complete translation Cyc can generate using its lexical knowledge of `Pump-TheWord` is one that drops the PP complement altogether:

```
(thereExists ?H
 (thereExists ?P
  (thereExists ?B
   (and
    (isa ?H Heart)
    (isa ?B Blood)
    (isa ?P PumpingFluid)
    (primaryObjectMoving ?P ?B)
    (providerOfMotiveForce ?P ?H)))))
```

In other words, the Cyclifier effectively parses the sentence as though it were "The heart pumps blood." To overcome this deficit, TextLearner does an analysis of the reified linkage for this sentence, which shows the following:

1) "pumps" is the main verb with "blood" as its object.

2) "to the lungs" is a prepositional complement, with "to" as the relevant preposition.

1) gives enough information for TextLearner to know that the semantic translation template for `Pump-TheWord` using the transitive NP comp frame is relevant. Thus is can use that template as the basis for building a new template that will allow it to handle the full verb phrase. **2** gives enough information for TextLearner to apply a common-sense based heuristic for interpreting the PP complement. In particular, the Cyc lexicon contains a mapping from English prepositions to "underspecified" CycL predicates that serve as semantic stubs for the interpretation of those prepositions. In this case, Cyc knows that `To-TheWord` maps to the predicate `to-UnderspecifiedLocation`. These underspecified predicates are extremely general, living at the top of a vast predicate hierarchy and have dozens (in some case hundreds) of specializations, each of which is a possible interpretation for any given use of a preposition. Cyc's strategy for building a new semantic translation template is to do a focused search, downward through the hierarchy of predicates under `to-UnderspecifiedLocation` to find the best possible predicate. To focus the search, Cyc applies commonsense knowledge about `PumpingFluid`, a possible denotation for `Pump-TheWord` already known to be applicable from the semantic translation template known to be relevant from **1)**.

`PumpingFluid` is the Cyc term that denotes the collection of all fluid-pumping events. To find a suitable role for "to," Cyc looks for a specialization of `to-UnderspecifiedLocation` that is the most-specific, required role for any fluid-pumping event. Since every fluid-pumping event is a location-change event, i.e.,

```
      (genls PumpingFluid Translocation)
```

and every location-change event has a location that is the "destination" of the object moving in the event, the role `toLocation` is required for any fluid-pumping event. This requirement is represented in Cyc as

```
(requiredActorSlots Translocation
toLocation)
```

As there are no more specific predicates in the `to-UnderspecifiedLocation` hierarchy that are also required, `toLocation` is selected. Thus Cyc combines the template for the transitive frame for "pump"

```
(and
(isa :ACTION PumpingFluid)
(providerOfMotiveForce :ACTION :SUBJECT)
(primaryObjectMoving :ACTION :OBJECT))
```

with a clause for handling the PP frame

```
(toLocation :ACTION :OBLIQUE-OBJECT)
```

to construct a new semantic translation template

```
Mt : GeneralEnglishMt
(verbSemTrans Pump-TheWord 311
(PPCompFrameFn DitransitivePPFrameType
   To-TheWord)
(and
(isa :ACTION PumpingFluid)
(providerOfMotiveForce :ACTION :SUBJECT)
(primaryObjectMoving :ACTION :OBJECT)
(toLocation :ACTION :OBLIQUE-OBJECT)))
```

This template allows the Cyclifier to produce a more complete parse that covers all of the components of the target sentence:

```
(thereExists ?PUMPING
(thereExists ?HEART
(thereExists ?BLOOD
(thereExists ?LUNGS
(and
 (isa ?PUMPING PumpingFluid)
 (isa ?HEART Heart)
 (isa ?LUNGS Lung)
 (isa ?BLOOD Blood)
 (primaryObjectMoving ?PUMPING ?BLOOD)
 (providerOfMotiveForce ?PUMPING ?HEART)
 (toLocation ?PUMPING ?LUNGS))))))
```

EBMT Templates

As noted in the previous section, one method by which TextLearner proposes semantics for input sentences is through the application of sentence-level templates. Due to some recent work in the area of Example-based Machine Translation (EBMT), TextLearner is now able to construct such templates by example, so that if a complete English-to-CycL mapping is presented, it can generalize the example into a template. EBMT traditionally has been used to effect translations from one natural language to another, by comparing large parallel corpora in different languages (Nagao 1984). TextLearner can acquire such templates by identifying high-scoring sentence-level interpretations and submitting the original English and the interpretation as an example mapping pair. Another method used is to extract a conjunction of provable clauses from an interpretation, the result of which is a high-scoring partial interpretation (high scoring because Cyc has proved it true), and submit this constructed interpretation along with the original English as an EBMT mapping example. For example, where the Cyclifier might return the following, partially correct CycL for "Angola is a country in Africa that has a long history of violence."

```
(thereExists ?STORY
 (thereExists ?ANGOLA
  (and
   (isa ?ANGOLA Country)
   (equals ?ANGOLA Angola)
   (inRegion ?ANGOLA ContinentOfAfrica)
   (isa ?STORY HistoricalNarrative)
   (duration ?STORY LongTime)
   (topicOf ?STORY ViolentAction)
   (hasRightsOver ?ANGOLA ?STORY))))
```

Though the translation of the entire sentence is unsatisfactory—it claims that there is some story about violence that Angola has rights over—there are parts of it, namely that *Angola is a country* and *Angola is in Africa* that Cyc can prove to be true given its existing knowledge of geopolitics. TextLearner in this case would pull both literals out and combine them, proposing this partial mapping that can be used as an EBMT template:

```
(thereExists ?ANGOLA
 (and
  (isa ?ANGOLA Country)
  (equals ?ANGOLA Angola)
  (inRegion ?ANGOLA ContinentOfAfrica)))
```

The EBMT code then generalizes the mapping into a template for rapidly interpreting sentences of the form

"COUNTRY is a TYPE in REGION that …" which would be sufficient from extracting at least the type and location of a geopolitical entity, even if the relative clause cannot be understood.

Conclusion

The TextLearner system is a prototype program that analyzes documents with the goal of deriving both their content as well as new rules for improving its own ability for deriving content. By generating a rich model of its target document, TextLearner is able to identify gaps in its understanding and hypothesize new rules for interpreting text. This paper has described four types of rule acquisition that TextLearner implements; the authors believe that this indicates a rich area for future Natural Language Processing research.

References

Bertolo, Stefano, *et al.*, "Quirk Final Report" 2000.

Burns, Kathy J. and Davis, Anthony B., "Building and maintaining a semantically adequate lexicon using Cyc," *The Breadth and Depth of Semantic Lexicons*, Evelyn Viegas, ed., Kluwer, 1999, pp. 121–143.

Curtis, Jon, G. Matthews, D. Baxter, "On the Effective Use of Cyc in a Question Answering System," *Proceedings of the IJCAI Workshop on Knowledge and Reasoning for Answering Questions*, Edinburgh, Scotland, July 30, 2005, pp. 61-70.

Curtis, Jon, D. Baxter, J. Cabral, "On the Application of the Cyc Ontology to Word Sense Disambiguation", *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, Melbourne Beach, Florida, May 11-13 2006, pp. 652-657.

Nagao, Makoto, "A Framework of Mechanical Translation between Japanese and English by Analogy Principle," in A. Elithorn and R. Banerji. *Artificial and Human Intelligence*. 1984.

Ramachandran, Deepak, P. Reagan, K. Goolsbey, "First-Orderized ResearchCyc: Expressivity and Efficiency in a Common-Sense Ontology", *Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications.* Pittsburgh, Pennsylvania, July 2005.

Temperely, Davy, D. Sleator, J. Lafferty, "Link Grammar" http://www.link.cs.cmu.edu/link/ December 2004.

Webber, Bonnie *A Formal Approach to Discourse Anaphora*, Doctoral Dissertation, Division of Applied Mathematics, Harvard University, 1978.

Witbrock, M., and Schneider, D., "Reading – Learning – Comprehension" DARPA proposal, 9/23/2004.