Resolving References and Identifying Existing Knowledge in a Memory Based Parser

Kevin Livingston and Christopher K. Riesbeck

EECS Department
Northwestern University
2133 Sheridan Rd. Evanston, IL 60208, USA
{livingston; c-riesbeck}@northwestern.edu

Abstract

Learning by reading systems, designed to acquire episodic (instance based) knowledge, ultimately have to integrate that knowledge into an underlying memory. In order to effectively integrate new knowledge with existing knowledge such a system needs to be able to resolve references to the instances (agents, locations, events, etc.) it is reading about with those already existing in memory. This is necessary to extend existing memory structures, and to avoid incorrectly producing duplicate memories.

Direct Memory Access Parsing (DMAP) leverages existing knowledge and performs reference resolution and memory integration in the early stages of parsing natural language text. By performing incremental memory integration our system can reduce the number of ambiguous sentence interpretations and coreference mappings it will explore in-depth, however this savings is currently canceled out by the run-time cost of reference resolution algorithm. This paper supports the continued investigation of this line of research, which is to identify and evaluate the extent to which semantic and episodic memory can facilitate natural language understanding, especially when used early in the language understanding process.

Introduction

Learning by reading has at its core the production and utilization of a large scale knowledge base, a memory, for the learning system to integrate what it is learning with what it already knows. Ultimately learning readers are acquiring knowledge from text.

There are two kinds of knowledge typically acquired by learning by reading systems. The first is generalized knowledge such as ontologies (e.g. the heart is a pump), and relations (e.g. the body is made of cells). The second is episodic or instance based knowledge, for example, Barack Obama is president, Mozart died in 1791. This type of knowledge also includes event instances, such as the Madrid train bombings in 2004, or the attack in Afghanistan last week. Our focus is primarily on acquiring and integrating episodic (instance based) knowledge.

For systems learning about specific instances, including event descriptions, it is important to be able to ground all references to existing structures in memory and create new structures when needed. This process also involves the resolution of coreferences across sentences. For example, consider a system reading the following two sentences in the context of a news story.

An attack occurred in Iraq.
Insurgents bombed a group soldiers.

In reading the second sentence, there are three references that need to be resolved in order to produce a complete understanding. These references are the insurgents, the soldiers, and a bombing event. A reader needs to understand these references within the context of the story, by establishing *coreference mappings* to other references in the same story, including that the bombing event the same as the attack in the first sentence, and ground these references to any relevant structures already in memory. Failure to do so would result in fragmented understanding, or replication of instances or events which does not correspond to the actual state of the world. For example, incorrectly assuming there are two distinct events in the two sentences above, would also mean not knowing that the bombing occurred in Iraq, or that the attack targeted soldiers.

We are interested in producing a learning by reading system that can read, for example, about an election taking place one day, and then read about the winner the next day, and recognize that the descriptions are referring to the same event. The system should be able to extend its existing representation of the election, produced from the first story, with the new knowledge of the outcome and winner of the election. Further, if weeks or months later the system reads a story discussing the same election, it should recall the same memory structure it constructed in the past, not hallucinate a new, nearly identical, event. Building complete understandings of large collections of events requires not only the ability to recognize repeated references to the same event, but also the ability to understand and learn how new events relate to old ones. For example, that the bombings performed today by Israel were in response to the kidnappings of Israeli soldiers yesterday by members of Hezbollah. To be able to do this a reader must be able to ground references in existing

memory structures (recognize what it already knows), and integrate new knowledge to extend what it already knows.

Direct Memory Access Parsing (DMAP) is an end-to-end language understanding model, running from input text through the production of references to existing knowledge structures in its underlying memory, with the ability to create and integrate new knowledge when necessary. In contrast to the standard syntactic-semantic pipeline that defers memory integration and reference resolution until the end, DMAP performs reference resolution and memory integration as it parses. A motivation for this algorithm is our belief that the grounded references and existing memory can be used to guide coreference decisions and overall story understanding. Our research is exploring the construction of a system that can use what it already knows to understand new information, and also rapidly recognize and understand information it has already seen. Fully integrating an interpretation in a large and incomplete memory presents numerous challenges, many of which are likely not unique to the DMAP approach.

Reference Resolution and Memory

DMAP uses a collection of language patterns that recursively map textual references to knowledge structures, and even specific instances in memory when possible. Instances include not only individuals like people, George W. Bush, or countries, Iraq, but also events in episodic memory, such as the Madrid train bombings of 2004. Functionally the pattern matcher in DMAP is similar to a tabular chart parser (Kay 1996). The pattern matcher has mappings from sequences of strings, lexical, and semantic concepts, to semantic assertions which they can be translated into. These assertions are grounded in the existing knowledge base whenever possible, and new assertions are generated only as needed. More detail on the architecture of this implementation of DMAP is presented by Livingston and Riesbeck (2007).

DMAP is an attempt to treat language understanding as a recognition and pattern matching process, as opposed to a knowledge construction and reasoning task. As sentences are being parsed, in order to build a coherent and complete representation of the story, DMAP establishes coreference mappings between the references in new sentences and those from previous sentences. As the example sentences in the introduction show, these coreference mappings are essential to building a complete representation of what is being described when a description spans multiple sentences. Identifying what references corefer within a story can produce a coherent semantic structure for that story, however those references still need to be grounded to existing references in memory to complete the understanding of that story and tie it to existing episodic knowledge.

Semantic References

DMAP translates text to semantic structure early in the understanding process. For example, the sentence "The soldiers attacked the bunker" produces the following semantic structure (represented in the semantics of ResearchCyc (Lenat 1995), the underlying knowledge base used by this implementation of DMAP).

```
(isa ?s (GroupFn Soldier))
(isa ?a AttackOnObject)
(performedBy ?a ?s)
(isa ?b Bunker)
(objectAttacked ?a ?b)
```

The representation is a set of predicate logic assertions. Symbols starting with a question mark represent an ungrounded, or open, reference. In a representation such as this, where events are reified, the event becomes another reference in the semantic structure just like the reference to the soldiers or the bunker. Our system constructs interpretations by identifying which of these references corefer to similar semantic references produced from the sentences that preceded this sentence in the story, and by grounding all of these references in the underlying memory. If no suitable reference can be identified in memory, DMAP will create a new instance to fill the role. Resolving references to memory includes not only objects, like people and places, but also events, like elections and bombings. For example, if text refers to the collapse of two buildings in New York, we would like to ground this reference to the existing memory structure for the September 11th attacks.

Identifying references is complicated in a large knowledge base (like ResearchCyc), by incompleteness, variations in level and scope of descriptions (e.g. describing a group of people as sailors or military personnel; or using the predicate `parentOf` vs. the predicate `fatherOf`), variations in how two similar events can be described by different sources, and also inconsistencies and noise in the knowledge. However, by far the most complicated problem in grounding event references in an underlying memory is operating under an open world assumption.

Maintaining an open world assumption means that some pieces of the interpretation produced from a text may be present in memory, while others may not. For example, imagine a specific event is known by a reader to have occurred in a location on a specific date, however the performer of that event is yet unknown. If a new story comes in stating all three pieces of information, querying for the conjunct of all three assertions will result in a failure to retrieve anything from memory (for the standard methods of handling conjuncts used by most reasoning systems and knowledge bases), since one piece of information is unknown the logical conjunct is also unknown.

Reference Resolution

To operate in an open world, our DMAP implementation breaks the representation into pieces and queries for them one at time, instead of querying for the entire representation constructed thus far for the story. Each piece corresponds to a set of assertions that were produced from one pattern matching rule. Patterns roughly correspond to one English phrase or sentence. Each sentence will therefore produce at least one such piece, but could produce more. The previous example sentence "The soldiers attacked the bunker," produces only one set of assertions. However, changing the sentence to "The soldiers attacked the bunker on July 18, 2008," will result in two sets of assertions, one corresponding to the pattern we have already discussed <agent> attack <object>, and a second corresponding to <event> on <date>. We refer to the process of applying patterns to translate text into sets of assertions, as *parsing* in our DMAP implementation. Our parsing stage only includes references to ground instances when they were explicitly mentioned in the text being read, for example "Iraq" or "the United States Army". Further, the only coreference information available after parsing is that which is explicitly encoded in the patterns, all other coreference resolution is performed in the subsequent reference resolution algorithm. For a more details of the DMAP parsing algorithm see Livingston and Riesbeck (2007).

The reference resolution algorithm receives as inputs the semantic interpretation of the story prior to the current sentence, including coreference mappings between the references in that interpretation, and a list of bindings to existing ground instances that interpretation could be referencing. It also receives the semantic fragments from parsing the current sentence. The algorithm's task is to output a new set of coreference mappings that integrate the new assertions, as well as a new list of ground instances to which those references could be referring. The algorithm may remove ground references from the list if they do not agree with the new information being added to the representation (e.g. the system could resolve which "George Bush" is being referred to, when previously it was ambiguous). New references to ground instances may be added as well. For example, if the text referrers to the terrorists that performed an attack, and the attack was already grounded as the Madrid train bombings of 2004, it can now ground the terrorists as members of Al-Qaeda.

When bindings (ground instances) are present for a given reference, and DMAP is attempting to decide if that reference corefers with a new reference, DMAP can substitute those bindings into the assertions where the new reference appears and then look to confirm this assertion in memory. For example, if DMAP is reading a story about a bombing, and has a reminding to a ground instance, `Bombing-54`, and the next sentence is, "The attack was performed by Al-Qaeda," DMAP needs to decide if the attack in this sentence corefers to the bombing in the story it is already tracking, and if so if they both then collectively still refer to `Bombing-54` in memory. The sentence in this example produces, among others the assertion `(performedBy ?attack Al-Qaeda)`. To see if the reference `?attack` corefers with the bombing in the story and `Bombing-54` at the same time, DMAP can substitute the value in and attempt to *confirm* the assertion `(performedBy Bombing-54 Al-Qaeda)` in memory. If this query returns true, then the system is can be confident in making `?attack` ("the attack") from this sentence corefer with the reference to the bombing, and that both likely refer to `Bombing-54`, unless contradicting information is later found when integrating subsequent sentences.

However, it is possible that the query for `(performedBy Bombing-54 Al-Qaeda)` could return no results. Since DMAP must operate under the open world assumption (as does, presumably, any open ended learning system), it does not know if this means `Bombing-54` is known to be performed by another agent, or if it is simply unknown who performed `Bombing-54`. To answer this question DMAP must issue an open, and more time consuming, query to find out if there is a known performer for `Bombing-54`. This can be accomplished by querying for `(performedBy Bombing-54 ?agent)`. If no results are returned, the performer is unknown, and it is *consistent* that `Bombing-54` could have been performed by `Al-Qaeda`. If, on the other hand, a value is returned, it must be checked to see if it is consistent with the value provided by the story. For example, the value of `?agent` could be a member of Al-Qaeda, or it could be a parent organization (assuming for the moment Al-Qaeda has one). Currently the reference is only seen as consistent if it is equal (in which case the original fully grounded query would have returned true) due to resource constraints. It is a source of future work to expand this to accept variability in level of description or metonymy, for example "hundreds" vs. "418" or "Iraq" vs. "Baghdad".

Finally if the open query returned a result that was not consistent with `Bombing-54`, two options are available to the DMAP implementation. The first is make the two references in question corefer anyway and discard the reminding to `Bombing-54`. The second is to keep the reminding to `Bombing-54`, and decide that the reference to "the attack" referrers to a different event that does not corefer with the first. DMAP will continue to search for confirmed results, potentially creating two event references here, if it can find confirmed remindings in memory which support this. However, in lieu of that, the system will prefer to produce an interpretation of the story that is more coherent, as is measured by how many assertions share arguments. Thus if `Bombing-54` was not a good match, it will continue to produce one single event description, and discard the reminding to `Bombing-54`, believing instead that the event in the story is some new event not previously encountered. These are not the only potential solutions available, just the ones used by this implementation currently, see future work for more discussion.

Experiment

We believe a language understanding system should be able to leverage its existing knowledge to improve understanding and decrease both the level of apparent ambiguity and runtime. The DMAP model maps text to semantic and episodic knowledge early in the language understanding process. Therefore we measured the difference in the runtime and the amount of ambiguity evaluated by DMAP when a story was seen for the first time, versus when that story's content was already in memory. Ideally DMAP would be able to read something it already knows faster than it could the first time.

The complexity of the reference resolution problem grows with the number of references in the story, since each new reference has to be integrated with the ever growing list of references up until that point. This obviously results in an exponential operation. Integrating these references with memory can further amplify this effect, especially when there is an ambiguity of reference to existing knowledge structures (which "President Bush" is being referred to, or which "attack in Afghanistan", etc.). To start exploring the impact ambiguity has on the reference resolution algorithm, we represented the same semantic and episodic content in three stories, varying the number of sentences and references DMAP needed to integrate in order to understand the story.

The following three stories produce identical interpretations and refer to the same single event.

Story A

An attack occurred in Afghanistan.
The bombing was performed by Al-Qaeda.
The attack occurred on July 18, 2008.
The attack targeted United States soldiers.

Story A presents an event description spread across four sentences. Each sentence contains a reference to an attack, or in the case of the second sentence, a bombing. When DMAP processes this story, it will read the first sentence and construct a representation for an attack in Afghanistan. It will also look in memory for candidate events to ground this reference in, and to use for subsequent reference resolution. With every subsequent sentence DMAP will have to decide if the event referenced in that sentence is the same as the event it is already tracking, using the reference resolution algorithm described earlier in this paper. DMAP will have to resolve four distinct event references.

Story B

There was an attack on July 18, 2008.
The bombing occurred in Afghanistan.
Al-Qaeda targeted United States soldiers.

Story B tells the same story as Story A, although it does so in three sentences. Story B has three distinct event references that need to be resolved. The first two sentences have an attack and a bombing explicitly mentioned as their subjects. The third sentence has an implicit reference to an event, which appears clearly in the semantic representation of targeting.

Story C

Al-Qaeda performed an attack on July 18, 2008.
United States soldiers were targeted by the bombing in Afghanistan.

Story C again represents information identical to that presented in Stories A and B. This story compresses the explanation into two sentences, and there are two event references that need to be resolved by DMAP.

All stories contain the same five pieces of information about the event: type of event, location, date, perpetrator, and target. The major difference is that Story A has four distinct references to the event that ultimately need to be found to corefer, Story B has three distinct references, and Story C only two references that need to be resolved inter-sentence.

Representation and References

Processing Stories A, B, and C result in identical sets of predicate logic assertions. Below is the predicate logic representation of Stories A, B, and C. The representation is in ResearchCyc semantics. Symbols prefixed by "DMAP" are instances created by DMAP in this reading, or in a previous reading and then retrieved from memory. Only assertions directly relevant to the interpretation of the text are presented by DMAP. For example, DMAP does not re-assert that `DMAP-TerroristGroup-162` (Al-Qaeda) is an instance of `TerroristGroup when that is already known in memory and not mentioned in the text.`

Representation of Stories A, B, and C

```
(isa DMAP-Bombing-14991 Bombing)
(eventOccursAt DMAP-Bombing-14991
               Afghanistan)
(dateOfEvent DMAP-Bombing-14991
  (DayFn 18 (MonthFn July (YearFn 2008))))
(performedBy DMAP-Bombing-14991
             DMAP-TerroristGroup-162)
(intendedAttackTargets DMAP-Bombing-14991
                       DMAP-Group-14990)
(isa DMAP-Group-14990
     (GroupFn MilitaryPerson))
(isa DMAP-Group-14990
     (GroupFn (MemberFn (ArmedForcesFn
               UnitedStatesOfAmerica))))
(isa DMAP-TerroristGroup-162 Agent-Generic)
(isa UnitedStatesOfAmerica Organization)
(isa Afghanistan PartiallyTangible)
(isa (DayFn 18 (MonthFn July (YearFn 2008)))
     CalendarDay)
```

Processing Story A will result in one set of assertions per sentence, each being processed by the reference resolution algorithm independently. This is a little different than what happens in the processing of Story C. For example, the

		Time						Ambiguity			
Ground Instances Available	Story	Pattern Matching (ms)	Change	Reference Resolution (ms)	Change	Total Time (ms)	Change	States	Change	Reference Mappings Explored	Change
No	A	680		90		770		45		27	
	B	390		30		420		18		8	
	C	2250		40		2290		18		15	
Yes	A	720	40 (106%)	200	110 (222%)	920	150 (119%)	45	0 (0%)	23	-4 (85%)
	B	260	-130 (67%)	50	20 (167%)	310	-110 (74%)	10	-8 (56%)	2	-6 (25%)
	C	2400	150 (107%)	80	40 (200%)	2480	190 (108%)	14	-4 (78%)	12	-3 (80%)

Table 1. Processing time for Stories A, B, and C, with and without ground instances available in memory.

second sentence of Story C will produce two sets of assertions, one representing a bombing that targeted US soldiers, and another representing a bombing that occurred in Afghanistan. Due to the way the pattern matching occurred, the pattern matcher can indicate that the bombing in both sets of assertions is the same event, resulting in effectively one event reference for the whole sentence. Although there is an identical set of predicate logic assertions passing through the reference resolution algorithm, there are fewer references to resolve in Story C, then there are in Story B, and fewer in Story B then there are in Story A.

Results

The results from reading the three stories before the related assertions existed in memory, and then again after the ground references were available in memory to retrieve are listed in Table 1. The table shows the times taken to perform pattern matching, and reference resolution, as well as the total time for both operations. The difference for each story, contrasted between the two conditions, is given in the delta columns. In addition to the time taken to perform these operations, the number of combinations of a new sentence with the existing interpretation(s) is given in the states column. In the last column the number of reference resolution mappings attempted when combining new semantic fragments is given.

Patterns are currently (and naively) allowed to match nearly all possible subsets of the sentences including seeing in the second sentence of Story C only the words "... soldiers were … in …" and interpreting that as a reference to soldiers being popular (since "in" can be interpreted as fashionable or popular). Obviously longer sentences, such as those in Story C, have the opportunity to produce far more of these fragmentary understandings, and thus an increase in parsing time. Story A produces approximately 300 semantic fragments, Story B only 150, and story C over 650 semantic fragments. There is a linear relationship between the number of pattern matching rules applied (semantic fragments produced) and the total parsing time; processing runs at about 3ms per match.

All of these fragments are handed off to the reference resolution algorithm to determine which ones fit best. Using the algorithm described above, and attempting to maximize text coverage, mitigates much of the ambiguity handed off to the reference resolution algorithm, as can be seen in the processing times.

The run time for the reference resolution algorithm is proportional to the product of the number of semantic ambiguities explored and the number of reference mappings explored. Reference resolution time is fairly fast in all cases before ground instances exist in memory. In processing Story A, DMAP evaluates more ambiguity than B or C, and this corresponds to increased run time as well. Although Stories B and C evaluate the same number of semantic ambiguities, story C spends more time figuring out how to resolve the references between sentences.

When references are available in memory, DMAP spends more time exploring each ambiguity, nearly twice as long. However, in the case of Stories B and C, DMAP is able to significantly reduce the number of ambiguities which need to be explored. The slowdown is somewhat mitigated by a more direct path though the search space, although DMAP still operated at a net loss with respect to run time. When DMAP had references in memory to aid its decision making, it could process story B and C correctly by evaluating fewer states. It used 44% less for story B, and 22% less for story C. These stories have more ambiguity packed into individual sentences and therefore pruning opportunities are more available and more beneficial than for story A.

Even though Story B is longer than Story C (more sentences, not more words), and more complicated than story A, it still process the least number of ambiguities, and does so in the least amount of time. Further in all three stories (less so in Story A), DMAP was able to leverage memory in order to reduce the number of ambiguities

which it explored. The bottom line is that having the right kind of guidance from memory can greatly assist reference resolution and ambiguity reduction.

Related Work

Memory based natural language understating techniques date back to Quillian's (1969) Teachable Language Comprehender (TLC), which used semantic connections to propose parsing possibilities first (e.g. what might "a doctor's patient" mean?) and then verify the syntax second. PHRAN (PHRasal Analyzer) (Wilensky and Arena, 1980) incorporated the idea of phrasal patterns to direct memory parsers. DMAP (Martin, 1990) went further. Where the other parsers described would construct interpretations, DMAP searched a hierarchical memory base for matching or similar semantic structures, making parsing a process of recognition. Indexed Concept Parsing (ICP) (Fitzgerald, 1994) also used memory based techniques.

Most NLU work however uses a pipeline model where language understanding is broken down into a sequence of operations, which typically include part of speech tagging and syntax analysis followed by semantic analysis. Systems such as Frail3 (Charniak and Goldman, 1988) started to extend the pipeline by using pragmatic axioms during the parsing process. Grosz et al. (1995) present work on producing coherent discourse structures over multiple sentences.

Information Extraction (IE) approaches use both statistical and phrase based parsing. While much of the IE work focuses on entity detection and relation extraction from isolated sentences (Etzioni et al 2005), there has been work to build case frames starting with PALKA (Kim and Moldovan, 1993) and AutoSlog (Riloff, 1993). Humphreys et. al. (1997) also focus on event coreference.

Future Work

The results in this paper show that DMAP can reduce the number of ambiguities it needs to explore by incrementally grounding references in memory. However, the number of references that need to be resolved by this algorithm impacts its running time. The more interconnected and less ambiguous the semantic structures being produced by the parser are, the faster reference resolution can be performed. Further if the system had access to larger patterns like scripts (Shank and Abelson 1977) matching them would result implicitly in reference resolution. For example, matching the following story with the classic restaurant script, would resolve the ambiguity of "he" in the third sentence.

John went to the restaurant.
The waiter came over.
He ordered a hamburger.

The script would map "he" as the agent ordering to John and not the waiter. An in-domain example would be the ability to recognize, for example, that the attack from one sentence is being performed in retaliation to the bombing from a different sentence, which would allow DMPA to better understand who are the likely actors or targets, as it would be specified in the script-level knowledge of the concept retaliation.

Adding more complicated patterns will not likely be free, as having larger organizing structures does not dictate what happens when multiple structures are in play simultaneously, or other similarly complicated configurations. Scripts are being used to resolve references and track events over sequences of news stories in information extraction systems such as Brussell (Wagner 2009). We feel that this is an important direction of future research, which will open new opportunities for progress toward the goal of complete text to integrated-memory (end-to-end) language understanding, operating in a time scale comparable (or faster) than human reading speed.

Other future work includes extending how DMAP can ground references in memory and extend existing knowledge. Repair strategies (Martin 1990) are what DMAP invokes when information in a story is close but does not directly correspond to what is in memory. Currently when this DMAP implementation sees a discrepancy its only strategy is to assume the information is new, and discard any references the story has produced to existing memory. Other options could include changing the underlying knowledge to correspond with the new information, or believing that the new information is in error. Related to this would be to introduce more reasoning into the process that checks if references in memory are consistent. This would include more reasoning about anaphora, especially metonymy.

Work to reduce the number of semantic interpretations coming out of the parsing algorithm, or to better prioritize the order of their processing could reduce load on the reference resolution algorithm, and also potentially reduce ambiguity that results in more parsing states. One option is to investigate the integration of feedback from a syntactic parser to identify more or less likely parses due to syntactic structure. Another option would be leverage context to activate, deactivate, or prioritize rules so that those that produce more relevant semantic structure to the current context will be produced first.

This paper supports the continued investigation of this line of research, which is to identify and evaluate the extent to which semantic and episodic memory can facilitate natural language understanding, especially when used early in the language understanding process.

Acknowledgements

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, and Leo Ureel. 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 (Forbus 1993).

References

Charniak, E., Goldman, R., 1988. A logic for semantic interpretation, *Proceedings of the 26th annual meet-ing on Association for Computational Linguistics*, p.87-94, June 07-10, 1988, Buffalo, New York.

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international Conference on World Wide Web* (New York, NY, USA, May 17 - 20, 2004). WWW '04. ACM Press, New York, NY, 100-110.

Fitzgerald, W. 1994. Building Embedded Conceptual Parsers. PhD thesis, Northwestern University.

Forbus, K. D., de Kleer, J. 1993 *Building Problem Solvers*, MIT Press, Cambridge, MA, 1993

Forbus, K. D., Riesbeck, C. K., Birnbaum, L., Livingston, K., Sharma, A., Ureel, L. 2007. Integrating Natural Language, Knowledge Representation and Reasoning, and Analogical Processing to Learn by Reading, *Twenty-Second Conference on Artificial Intelligence* (AAAI-07), Vancouver, British Columbia, Canada, July 22-26, 2007

Grosz, B., Weinstein S., Joshi, A. 1995 Centering: A framework for modeling the local coherence of dis-course Computational Linguistics.

Humphreys, K., Gaizauskas, R., Azzam, S. 1997. Event coreference for information extraction. In *Proceedings of the Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, Madrid, Spain (1997) 75–81

Kay, M. 1996 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.

Kim, J. and Moldovan, D.. 1993 Acquisition of Semantic Patterns for Information Extraction from Corpora. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*, pages 171–176, Los Alamitos, CA,. IEEE Computer Society Press.

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

Livingston, K., and Riesbeck, C. K. 2007. Using Episodic Memory in a Memory Based Parser to Assist Machine Reading, *Working notes of the AAAI 2007 Spring Symposium on Machine Reading*, Stanford University, California, USA, March 26-28, 2007

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

Quillian, M. R. 1969. The teachable language comprehender. BBN Scientific Report 10. Bolt Beranek and Newman, Boston MA..

Riesbeck, C. and Martin, C., 1985 Direct Memory Access Parsing, Yale University Report 354, 1985.

Riloff, E. 1993 Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816. AAAI Press/The MIT Press.

Schank, R and Ableson, R 1977. *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Erlbaum 1977

Wagner, E. J., Liu, J., Birnbaum, L., Forbus, K. D.: 2009. Rich Interfaces for Reading News on the Web. In *Proceedings of the 2009 International Conference on Intelligent User Interfaces*. Sanibel Island, FL, 2009.

Wilensky, R., and Arens, Y.. 1980. PHRAN: A Knowledge-Based Natural Language Understander. In *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, PA. June 1980.