# A Trainable Message Understanding System*

**Amit Bagga**
**Joyce Yue Chai**
Department of Computer Science
Box 90129, Duke University
Durham, NC 27708-0129
Internet: {amit, chai}@cs.duke.edu

## 1 Introduction and Background

The Message Understanding Conferences (MUCs) have given a great impetus to research in information extraction (IE). The systems which have participated in the MUCs have been quite successful at extracting information from the domains that they have been trained on (MUC-4, 1992), (MUC-5, 1993), (MUC-6, 1995). The precision and recall statistics were around 60% and 50% respectively for MUC-6. However, these systems are domain dependent and customizing them to a new domain is a long and tedious process. For example, porting BBN's PLUM system from the Joint Ventures (MUC-5) domain to the Microelectronics (MUC-5) domain took approximately 3 person weeks (Weischedel, 1993). Moreover, training and adapting these systems to a particular domain is done by a group of computational linguists. These linguists determine all the ways in which the target information is expressed in a given corpus and then think of all the plausible variants of these ways, so that appropriate regular patterns can be written.

The explosion in the amount of free text material on the Internet, and the use of this information by people from all walks of life, has made the issue of generalized information extraction a central one in Natural Language Processing. Many systems, including ones from NYU (Grishman, 1995), BBN (Weischedel, 1995), SRI (Appelt, 1995), SRA (Krupka, 1995), MITRE (Aberdeen, 1995), and the University of Massachusetts (Fisher, 1995), have taken steps to make the process of customizing a system for a particular domain an easy one. Appelt et al. write, "If information extraction systems are going to be used in a wide variety of applications, it will ultimately be necessary for the end users to be able to customize the systems themselves in a relatively short time." (Appelt, 1995)

We have built a system that attempts to provide *any user* with the ability to efficiently create and customize, for his or her own application, an information extraction system with competitive precision and recall statistics. This paper will present the theory of the system and some details of an implementation. It will also describe a test of the system in which a 3 hour training session produced precision and recall statistics in the 60% levels and above.

*Supported by Fellowships from IBM Corporation.

## 2 System Architecture

As illustrated in Figure 1, there are three main stages in the running of the system: the Training Process, Rule Generalization, and the Scanning Process. During the Training Process, the user, with the help of a graphical user interface (GUI), takes a *few* prototypical articles from the domain that the system is being trained on, and creates rules (patterns) for the target information contained in the training articles. These rules are specific to the training articles and they are generalized so that they can be run on other articles from the domain. The Rule Generalization routines, with the help of WordNet [1], generalize the specific rules generated by the Training Process. The system can now be run on a large number of articles from the domain (Scanning Process). The output of the Scanning Process, for each article, is a semantic network (Quillian, 1968) for that article which can then be used by a Postprocessor to fill templates, answer queries, or generate abstracts.

### 2.1 Tools Used By the System

#### 2.1.1 IBM and Local Dictionaries

The system uses IBM's LanguageWare English Dictionary, IBM's Computing Terms Dictionary, and a local dictionary of our choice. The IBM dictio-

[1] WordNet is an on-line lexical reference system developed by George Miller and his group at Princeton University.



Amit Bagga and Joyce Yue Chai (1997) **A Trainable Message Understanding System.** In T.M. Ellison (ed.) *CoNLL97: Computational Natural Language Learning,* ACL pp 1-8.

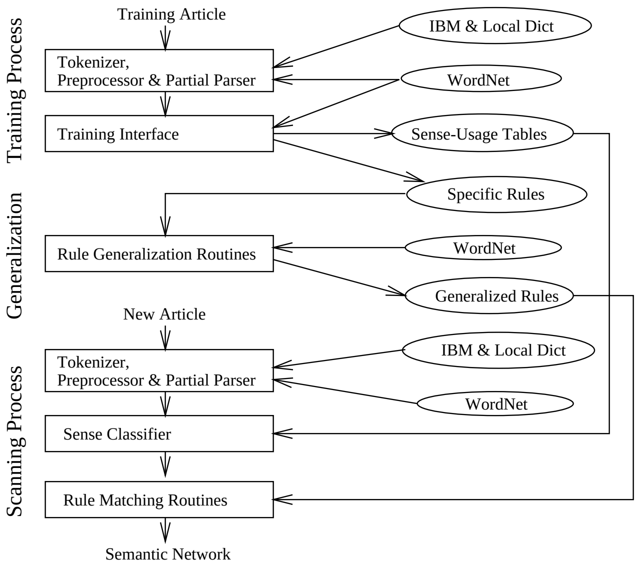


Figure 1: The Architecture

naries contain about 150,000 words while the local dictionary contains about 100 words.

#### 2.1.2 Gazetteer

The system also uses a gazetteer consisting of approximately 250 names of cities, states, and countries.

#### 2.1.3 WordNet

The system also uses WordNet (Miller, 1990). WordNet is an on-line lexical reference system in which English nouns, verbs, and adjectives are organized into synonym sets (*synsets*), each representing one underlying lexical concept (meaning or sense). Different relations link these synsets. WordNet contains approximately 95,600 word forms organized into some 70,100 synsets (Miller, 1990).

Consider the following example from (Miller, 1990): the synsets {*board, plank*} and {*board, committee*} each serve as unambiguous designators of two different meanings (or senses) of the noun *board*. Also associated with each synset is a short English description (gloss) of the meaning expressed by the synset. So the synset {*board*} (a person's meals, provided regularly for money), consisting only of the noun *board* itself, can be distinguished from the other senses by the gloss associated with it.

In addition, WordNet attempts to organize different senses of a word based on the frequency of usage of the senses. For example, a listing of all the synonyms of *board* yields 8 different synsets, each designating a different sense of the noun. The most commonly used sense (Sense 1) is {*board, plank*} while the least commonly used sense (Sense 8) is {*board*} (a flat portable surface (usually rectangular) designed for board games).

An important relationship present in WordNet, that is used extensively by our system, is the hyponymy/hypernymy (the subset/superset, or the ISA) relationship. A concept represented by the synset $\{x,\ x_1,\ \ldots\}$ is said to be a **hyponym** of the concept represented by the synset $\{y,\ y_1,\ \ldots\}$ if native speakers of English accept sentences constructed for such frames as *An x is a (kind of) y*. If this holds then we can also say that $\{y,\ y_1,\ \ldots\}$ is a **hypernym** of $\{x,\ x_1,\ \ldots\}$. For example, {*maple*} is a hyponym of {*tree*}, and {*tree*} is a hyponym of {*plant*}. Hyponymy is transitive and asymmetrical, and it generates a hierarchical semantic structure in which a hyponym is said to be below its superordinate.

More details about WordNet can be found in (Miller, 1990a).

### 2.2 The Tokenizer

The Tokenizer accepts ASCII characters as input and produces a stream of tokens (words) as output. It also determines sentence boundaries.

### 2.3 The Preprocessor

The Preprocessor accepts the tokens produced by the Tokenizer as input and produces a "word" stack as output. While creating the "word" stack, the preprocessor, using finite-state rules, tries to identify some important entities, like names of companies, proper names, etc., contained in the article. Groups of words that comprise these entities are collected together and put in one slot in the "word" stack. They are considered as one item for all future processing.

The Preprocessor identifies the following entities:

- cities, states, and countries
- names of companies
- software packages
- e-mail and Web addresses
- file and directory names
- dates, times, dollar amounts, telephone numbers, and Zip codes
- proper names.

### 2.4 Partial Parser

The Partial Parser accepts the word stack produced by the Preprocessor as input and produces a sequence of non-overlapping phrases as output. These phrases are stored in a phrase stack. The headword of each phrase is also identified and stored on the phrase stack. The parser recognizes noun groups, verb groups and preposition groups.

The Partial Parser is a finite-state parser and is largely borrowed from SRI's FASTUS system (Hobbs, 1993). The parser uses 14 finite-state rules to identify the noun groups, 7 rules to identify the verb groups, and one rule to identify the preposition groups. The last words of each group are identified as their headwords.

The output of the parser, the phrase stack, is used by both the Training and the Scanning processes.

### 2.5 The Training Interface

There are two parts to the Training Process: identification of the (WordNet) sense usage of headwords of interest, and the building of specific rules. Training is done by a user with the help of a graphical user Training Interface. Figure 2 shows a snapshot of the Training Interface. The Training Interface takes as input the phrase stack produced by the Partial Parser. Sense usage tables, and a collection of specific rules are built as a result of the Training Process.

#### 2.5.1 Identification of Sense Usage

The Training Process yields a collection of training-article-specific rules which are then generalized by the Rule Generalization routines. These generalization routines make use of the WordNet hypernym relationship by replacing the headwords present in a rule with one of its more general superordinates in the WordNet hierarchy. For this to be done, the Rule Generalization routines must know the sense (concept) of usage of the headword in the article it appears in, because the WordNet hierarchy is sense dependent. Moreover, training articles often contain headwords that are not used in their most frequent sense. For example, in a domain which advertises job openings, the noun *opening* will most likely be used as an "opportunity for employment or promotion" (Sense 4), rather than the most commonly occuring sense: "an open or empty space in or between things" (Sense 1). Therefore, for each of the headwords of interest, the user, based on the gloss associated with the senses provided by WordNet, has to decide which sense is being used in the article. If the user does not train on a particular headword, then, by default, it is assumed to have been used in its most commonly occuring sense.

The system, for each headword appearing in the article, keeps a count of the frequency of occurrence of the senses associated with it. All this information is stored in the form of a table (Sense-Usage Table) which is later used by the Scanning Process. During the Scanning Process, the system does not have the help of the user to determine what sense of a particular headword is being used. Neither can the system simply assume the most commonly occuring sense. Therefore, the Sense Classifier determines the senses of the headwords of the phrases based on the Sense-Usage Table built during the Training Process. The sense identified by the Sense Classifier is then used for all future processing.

#### 2.5.2 Building the Specific Rules

The user builds the collection of rules by actually building semantic networks for the training articles using the Training Interface. Specifically, the user scans the phrase stack one entry at a time and selects phrases that he or she feels should be translated to the output semantic network. Then the selected phrases are translated to nodes or transitions of the network using GUI provided operations. There are two operations used to build the semantic network: the ADD_NODE operation and the ADD_RELATION operation.

The ADD_NODE operation allows the user to add a node to the semantic network while the ADD_RELATION operation allows the user to add a transition between two nodes. For the ADD_RELATION operation, if either (or both) of the two nodes do not exist, the ADD_NODE operation is automatically invoked and the node is added to the network. Since the ADD_RELATION

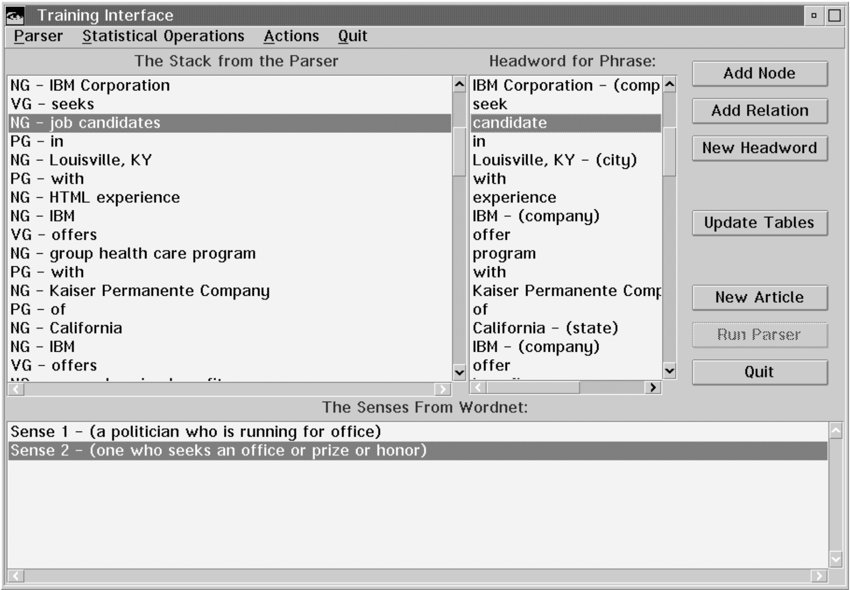


Figure 2: Snapshot of the Training Interface

operation subsumes the ADD_NODE operation, we only consider the ADD_RELATION operation. The system automatically creates and saves an ADD_RELATION rule for each ADD_RELATION operation performed by the user during training.

The user executes an ADD_RELATION operation by identifying the two objects (nodes) and the relationship (transition) between them. For example, consider the phrase stack of the sentence "IBM Corporation seeks job candidates in Louisville, KY with HTML experience." as shown in Figure 2. Suppose the user executes an ADD_RELATION operation by identifying *IBM Corporation*, and *job candidates* as the two nodes, and *seeks* as the transition connecting the nodes. The rule corresponding to this operation is shown in Figure 3. The left hand side (LHS) of the rule specifies the conditions that need to be satisfied for the RHS to be executed. For each rule, there are three conditions in the LHS. Each condition is a 4-tuple consisting of the following fields: the headword of the phrase, the type of phrase, the WordNet sense number identified by the user (default is 1), the headword "type" identified by the Preprocessor (default is "other_type"). The three conditions present in the LHS of the rule need not appear contiguously in a sentence of the article.

Training a 3000 byte article (approximately 1 page) takes approximately 10 minutes. The number of articles that the system must be trained on depends on the domain and the user's expectations of precision and recall. The more you train the system, the better the precision and recall. We are currently working on the problem of trying to predict the right number of articles on which the system must be trained, for a particular domain, to obtain target recall and precision statistics.

# 3 Generalization

Rules created as a result of the Training Process are very specific and can only be applied to exactly the same patterns as the ones present during the training. In order to make the specific rules applicable to a large number of unseen articles in the domain, a comprehensive generalization mechanism is necessary. We are not only interested in the generalization itself, but also in the strategy to control the degree of generalization for various applications in different domains.

## 3.1 Degree of Generalization

The hierarchical organization of WordNet (Miller, 1990) provides the possibility of automatic rule generalization of the rules. Philip Resnik has done some work earlier in using the hierarchical structure of WordNet (Resnik, 1995a) (Resnik, 1995b). With the large amount of information on semantic classification and taxonomy provided in WordNet, many ways of incorporating WordNet's semantic features with generalization are foreseeable. Although, at this stage, we only concentrate on the Hypernym/Hyponym feature.

¿From the training process, the specific rules contain three entities on the LHS as shown in Figure 3. Each entity is a quadruple, in the form of

[IBM Corporation, NG, 1,company], [seek, VG, 1, other_type], [candidate, NG, 2, other_type]
$\longrightarrow$ ADD_NODE(IBM Corporation), ADD_NODE(candidate),
ADD_RELATION(seek, IBM Corporation, candidate)

Figure 3: A Sample Rule

$(w_1, c_1, s_1, t_1)$, $(w_2, c_2, s_2, t_2)$,$(w_3, c_3, s_3, t_3)$
$\longrightarrow$ ADD_NODE($w_1$), ADD_NODE($w_3$), ADD_RELATION($w_2$, $w_1$, $w_3$)

Figure 4: An Abstract Specific Rule

$sp = (w, c, s, t)$, where $w$ is the headword of the trained phrase, $c$ is the part of the speech of the word, $s$ is the sense number representing the meaning of $w$, $t$ is the semantic type identified by the preprocessor for $w$. An abstract specific rule is shown in Figure 4.

For each $sp = (w, c, s, t)$, if $w$ exists in WordNet, then there is a corresponding synset in WordNet. The hyponym/hypernym hierarchical structure provides a way of locating the superordinate concepts of $sp$. By following additional hypernyms, we will get more and more generalized concepts and eventually reach the most general concept, such as {*person, human being,...*}. Based on this scenario, for each concept, different degrees of generalization can be achieved by adjusting the distance between this concept and the most general concept in the WordNet hierarchy. The function to accomplish this task is *Generalize(sp,h)*, which returns a synset list $h$ levels above the specific concept represented by $sp$ in the hierarchy. An example is shown in Figure 5.

sp = (IBM Corporation, NG, 1, company)
$\downarrow$ generalized at degree 1
Generalize(sp, 1) = {business, concern}
$\downarrow$ generalized at degree 2
Generalize(sp, 2) = {enterprise}
$\downarrow$ generalized at degree 3
Generalize(sp, 3) = {organization}
$\downarrow$ generalized at degree 5
Generalize(sp, 5) = {group, social group}

Figure 5: Degrees of Generalization for a Specific Concept

### 3.2 Generalized Rules

The process of generalizing rules consists of replacing each $sp = (w, c, s, t)$ in the specific rules by a more general superordinate synset from its hypernym tree in WordNet by performing the $Generalize(sp, h)$ function. The degree of generalization for rules varies with the variation of $h$ in $Generalize(sp, h)$. For example, Figure 6 shows the rule in Figure 3 generalized to two different degrees.

Figure 7 shows an abstract generalized rule. The $\subseteq$ symbol signifies the subsumption relationship. Therefore, $a \subseteq b$ signifies that $a$ is subsumed by $b$, or, in WordNet terms, concept $b$ is a superordinate concept of concept $a$. The generalized rule states that the RHS of the rule gets executed if *all* of the following conditions hold:

- A sentence contains three phrases (not necessarily contiguous) with headwords $W_1$, $W_2$, and $W_3$.
- The quadruples corresponding to these headwords are $(W_1, C_1, S_1, T_1)$, $(W_2, C_2, S_2, T_2)$, and $(W_3, C_3, S_3, T_3)$.
- The synsets, in WordNet, corresponding to the quadruples, are subsumed by *Generalize(*$sp_1$*,* $h_1$*)*, *Generalize(*$sp_2$*,* $h_2$*)*, and *Generalize(*$sp_3$*,* $h_3$*)* respectively.

## 4 Scanning New Articles

The goal of generalizing the rules is to generate semantic networks for unseen articles. The semantic networks are built with the help of the ADD_NODE and the ADD_RELATION operations present in the RHS of the rules. The Scanning Process consists of the following steps:

- Parse the unseen article and segment it into phrases belonging to one of NG, VG, or PG ($C_i$).
- Identify the headword ($W_i$) for each phrase.

- [{enterprise}], [seek, VG, 1, other_type], [{applicant}]
  $\longrightarrow$ ADD_NODE({enterprise}), ADD_NODE({applicant}),
  ADD_RELATION(seek, {enterprise}, {applicant})

- [{organization}], [seek, VG, 1, other_type], [{person}]
  $\longrightarrow$ ADD_NODE({organization}), ADD_NODE({person}),
  ADD_RELATION(seek, {organization}, {person})

Figure 6: Specific Rule in General Forms

$(W_1, C_1, S_1, T_1) \subseteq Generalize(sp_1, h_1), (W_2, C_2, S_2, T_2) \subseteq Generalize(sp_2, h_2),$
$(W_3, C_3, S_3, T_3) \subseteq Generalize(sp_3, h_3)$
$\longrightarrow$ ADD_NODE($W_1$), ADD_NODE($W_3$), ADD_RELATION($W_2$,$W_1$, $W_3$)

Figure 7: Generalized Rule

- Use the Preprocessor to identify the type ($T_i$) for each headword.
- Use the Sense Classifier (as described in Section 2.5.1 to assign the appropriate sense ($S_i$) to each headword.
- Each phrase can now be uniquely represented by $(W_i, C_i, S_i, T_i)$. Match $(W_i, C_i, S_i, T_i)$ with the LHS of a generalized rule.
- If the three entities $[Generalize(sp_i, h_i)]$ subsume three phrases $[(W_i, C_i, S_i, T_i)]$, within a single sentence in the article, the rule is fired and the RHS of the rule executed.

If we train on *IBM Corporation seeks job candidates* and generate the rule as in Figure 3, Table 1 lists some sentences that can be processed as the degree of generalization.

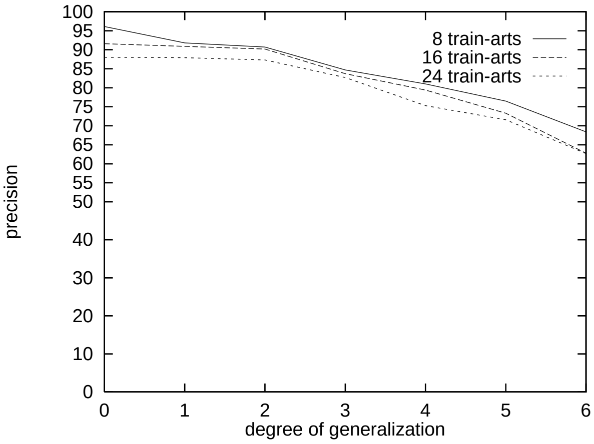


Figure 8: Precision vs. Degree of Generalization

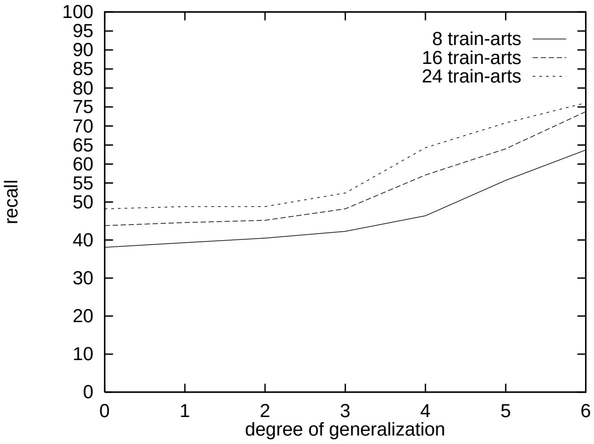


Figure 9: Recall vs. Degree of Generalization

## 5 Experiments

We designed an experiment to investigate how training and the generalization strategy affect meaning extraction. We trained our system on three sets of articles from the *triangle.jobs* USENET newsgroup, with emphasis on the following seven facts:

- Company Name. Examples: IBM, Metro Information Services, DCR Inc.
- Position/Title. Examples: programmer, financial analyst, software engineer.
- Experience/Skill. Example: 5 years experience in Oracle.
- Location. Examples: Winston-Salem, North Carolina.
- Benefit. Examples: company matching funds, comprehensive health plan.

| degree | Noun Phrase | Verb Phrase | Noun Phrase |
|---|---|---|---|
| 0 | GTE (any company) | seeks, looks for, searches | job candidates |
| 1 | Auction Agency (any business) | seeks, looks for, searches | bidder |
| 2 | Motor Factory (any enterprise) | seeks, looks for, searches | engineers |
| 3 | Police (any organization) | seeks, looks for, searches | the fugitive (any person) |
| 4 | Biology Lab (any group) | seeks, looks for, searches | missing frog (any life form) |

Table 1: Sample Sentences that Can Be Processed in the Scanning Part

- Salary. Examples: $32/hr, 60K.
- Contact Info. Examples: Fax is 919-660-6519, email address.

The first training set contained 8 articles; the second set contained 16 articles including the first set; and the third set contained 24 articles including those in the first two sets. For rules from each training set, seven levels of generalization were performed. Based on the generalized rules at each level, the system was run on 80 unseen articles from the same newsgroup to test its performance on the extraction of the seven facts.

The evaluation process consisted of the following step: first, each unseen article was studied to see how many facts of interest were present in the article; second, the semantic transitions produced by the system were examined to see if they correctly caught any facts of interest. The precision and recall curves with respect to the degree of generalization are shown in Figures 8 and 9 respectively.

In the precision vs. degree of generalization graph (Figure 8), precision decreases from 96.1% to 68.4% for the first training set as the degree of generalization increases from 0 to 6. The first set of eight training articles has better performance on precision. The fact that precision decreases with increased numbers of training articles seems to be counter intuitive initially. But, as the number of training articles increase, the the number of rules increase; which increases the chance that some piece of irrelevant information may trigger one of the rules, thereby decreasing the precision. In the recall vs. degree of generalization graph (Figure 9), for the third training set of 24 articles, recall increases from 48.2% to 76.1% as generalization degree increases. As expected, the third training set out-performed the other two training sets on recall.

In Figure 9, there is a jump in recall as we go from generalization degree 3 to generalization degree 4. This gives rise to the following important question: Why does a certain degree of generalization have a big impact on extracting a fact(s)? Moreover, with the increase in the degree of generalization, precision tends to fall while recall tends to increase. The question that arises here is: What degree of generalization gives us the best compromise between precision and recall? We are currently conducting further research that will help us answer such questions.

## 6 Conclusion

This paper describes a trainable system for meaning extraction. The critical parts in the system are the preprocessor, the partial parser, the training interface, the rule interpreter, the rule generalization routines, and the rule matching routines.

Our system allows a person to train a small number of texts from a particular domain, to get the desired information from a larger corpus of texts. The training effort is reduced to a few hours and the person training the system need not be a linguist or domain expert.

## 7 Acknowledgment

We wish to thank Jerry Hobbs of SRI for providing us with the finite-state rules for the parser.

We also wish to thank our advisor Dr. Alan W. Biermann for all his help, advise, and support.

## References

Aberdeen, John, et al. 1995. MITRE: Description of the *ALEMBIC* System Used for MUC-6, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 141-155, November 1995.

Appelt, Douglas E., et al. 1995. SRI International: Description of the FASTUS System Used for MUC-6, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 237-248, November 1995.

Fisher, David, et al. 1995. Description of the UMass System as Used for MUC-6, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 127-140, November 1995.

Grishman, Ralph. 1995. The NYU System for MUC-6 or Where's the Syntax? *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 167-175, November 1995.

Hobbs, J., et al. 1995. FASTUS: A system for Extracting Information from Text, *Human Language Technology*, pp. 133-137, 1993.

Krupka, George R. 1995. Description of the SRA System as Used for MUC-6, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 221-235, November 1995.

Miller, G.A. 1990. Introduction to WordNet: An On-Line Lexical Database. *WordNet Manuals*, pp. 10-32, August 1993.

Miller, G.A., et al. 1990a. *Five Papers on WordNet*, Cognitive Science Laboratory, Princeton University, No. 43, July 1990.

*Proceedings of the Fourth Message Understanding Conference (MUC-4)*, June 1992, San Mateo: Morgan Kaufmann.

*Proceedings of the Fifth Message Understanding Conference (MUC-5)*, August 1993, San Mateo: Morgan Kaufmann.

*Proceedings of the Sixth Message Understanding Conference (MUC-6)*, November 1995, San Francisco: Morgan Kaufmann.

Quillian, M. Ross. 1968 Semantic Memory, In *Semantic Information Processing*, M. Minsky (ed.). Cambridge, Massachusetts: MIT Press, 1968, pp. 216-270.

Resnik, Philip. 1995a Using Information Content to Evaluate Semantic Similarity in a Taxonomy, *Proceedings of IJCAI-95*, 1995.

Resnik, Philip. 1995b Disambiguating Noun Groupings With Respect to WordNet Senses, *Third Workshop on Very Large Corpora*, 1995.

Weischedel, R., et al. 1993. BBN: Description of the PLUM System as Used for MUC-5, *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pp. 93-107, August 1993.

Weischedel, Ralph. 1995. BBN: Description of the PLUM System as Used for MUC-6, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 55-69, November 1995.