Building a Machine Learning Based Text Understanding System

Stephen G. Soderland
Department of Radiology, UCLA
Los Angeles, CA 90024

Abstract

Text understanding systems are approaching the point of being a practical technology as long as the system is trained for a narrowly defined domain. Machine learning and statistical approaches can minimize the effort involved in adapting a text understanding system to a new domain.

This paper presents a system whose goal is deep understanding, limited only by the necessity of designing a formal representation of the target concepts relevant to the domain. This system is an advance over previous machine learning based systems because of its richer output representation, and an advance over equally expressive text understanding systems because of its more extensive use of machine learning.

1 Information Extraction from Free Text

A variety of systems have been developed in recent years that extract information from text. None of them attempts general-purpose understanding, but instead focus on narrowly defined information needs. A domain is defined as a collection of documents and a clear definition of the target concepts for that domain. Some systems look for fairly simple types of information such as scanning seminar announcements for time, location, and speaker [Freitag, 1998], on-line job postings for position title, location, skills needed, and salary [Califf and Mooney, 1997], or rental ads for number of bedrooms, price, and neighborhood [Soderland, 1999]. Everything else in the document is ignored as irrelevant.

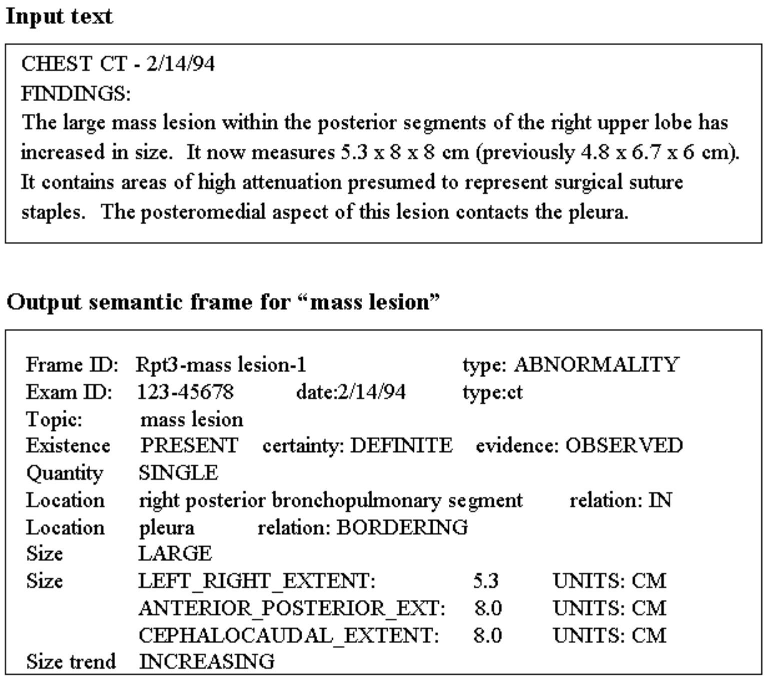
Information systems that aim for a deeper understanding have come out of a series of DARPA sponsored Message Understanding Conferences. These systems must still be adapted to one domain at a time, but have a richer representation of the information relevant to the domain. One example is the terrorism domain from the MUC-4 conference [1992], which consisted of news stories about Latin American terrorism and represented the target concepts with an output template for each specific event with details about the perpetrators, victims, and physical targets. Another example is the joint ventures domain of the MUC-5 conference [1993] where the domain consisted of news stories about joint business ventures and the target concept was details about the companies involved, the business activity, and the ownership of the joint venture.

Work on medical text understanding systems also aim at deep understanding. Examples of this are the work of Friedman et al. [1997, 1994] and of Taira and Soderland [2001], both groups initially developing their systems in the domain of radiology reports. The target concept is the findings mentioned in the report and details about their properties: such as existence, size, location, clinical trend, and interpretation. The assumption is that *everything* in a radiology report is relevant information, so the output representation must be deep enough to capture as much of the detail and nuances as possible.

Figure 1 shows an example from Taira and Soderland's system, a portion of a radiology report and output for one of the findings mentioned, a mass lesion in the right upper lobe of the lung. Properties extracted are the existence (present), quantity, location, size, and size trend (increasing). Note that some field values, shown in all capital letters, have been converted to a normalized form, and the anatomical locations that have been replaced with preferred terms from a medical controlled vocabulary.

How close is all of this to what could be called deep understanding of the text? The limit of these systems is primarily the need to explicitly design an output representation in advance. The system can only map free text to a formal representation to the extent that the representation can express the concepts in the text. Anything else in the text must necessarily be ignored. Thus in the terrorism domain, if a story mentions the color of the get away car or the political affiliations of a kidnapping victim, these will be ignored. In the joint ventures domain, the name of the company spokesman is ignored. If these concepts were considered important, the system designers would need to include *color-of-car* or *spokesman* as part of the output representation.

The two main challenges for automatic text understanding are creating a sufficiently expressive output representation for the target concepts of a domain, and then acquiring the domain-specific knowledge of how those concepts are represented in free text. This knowledge may include a



Input text

CHEST CT - 2/14/94
FINDINGS:
The large mass lesion within the posterior segments of the right upper lobe has increased in size. It now measures 5.3 x 8 x 8 cm (previously 4.8 x 6.7 x 6 cm). It contains areas of high attenuation presumed to represent surgical suture staples. The posteromedial aspect of this lesion contacts the pleura.

Output semantic frame for "mass lesion"

```
Frame ID:   Rpt3-mass lesion-1                     type: ABNORMALITY
Exam ID:    123-45678        date:2/14/94          type:ct
Topic:      mass lesion
Existence   PRESENT    certainty: DEFINITE    evidence: OBSERVED
Quantity    SINGLE
Location    right posterior bronchopulmonary segment       relation: IN
Location    pleura        relation: BORDERING
Size        LARGE
Size        LEFT_RIGHT_EXTENT:            5.3     UNITS: CM
            ANTERIOR_POSTERIOR_EXT:       8.0     UNITS: CM
            CEPHALOCAUDAL_EXTENT:         8.0     UNITS: CM
Size trend  INCREASING
```

Figure 1. A portion of a radiology report and one of the semantic frames extracted from it.

glossary with word senses appropriate to the domain, patterns of syntactic word attachments, and semantic patterns of ways that target concepts are expressed. Efficient methods of knowledge acquisition are needed if text understanding systems are to be of practical use.

Human effort can be minimized with a domain-independent system that uses machine learning and statistical techniques to acquire domain-specific knowledge. A system developer or end user creates sets of hand-tagged training examples from which the system automatically derives syntactic and semantic patterns. This is less labor intensive than creating complex rules by hand and can capture the subtlety and wide variation found in natural language. Some of the shallower extraction systems are entirely machine learning based. Freitag [1998] uses a combination of machine learning classifiers; Califf and Mooney [1997], Soderland [1999], and Freitag and Kushmerick [2000] use a variety of rule learning methods; Freitag and McCallum [2000] use hidden Markov models. Each of these systems finds extraction patterns in a single step with no explicit syntactic parsing, and operate best on semi-structured text rather than free narrative. Ciravegna [2000] obtains good results by learning a set of rules, then augmenting and correcting the rules in a multi-step system.

The medical text understanding system of Friedman et al. [1997, 1994] uses a manually constructed semantic grammar specifically designed for the domain of radiology reports. Most systems developed for the MUC conferences use considerable hand coding of domain knowledge, such as syntactic/semantic extraction patterns. One MUC system that used machine learning was that of the University of Massachusetts [Lehnert et al., 1993] which derived extraction rules from training. BBN [Miller et al., 1998] went a step further and trained a statistical model to handle both full syntactic parsing and identifying semantic relations between constituents in the parse tree. Both systems used domain-specific code to filter, merge, and format output from the machine learning system.

The work presented here combines extensive use of machine learning with an expressive output representation that moves information extraction in the direction of deep understanding. The remainder of this paper describes the system and discusses issues in designing an output representation and in learning semantic interpretation rules to map free text into that representation.

2 Steps in Text Analysis

Our text understanding system has six main steps:

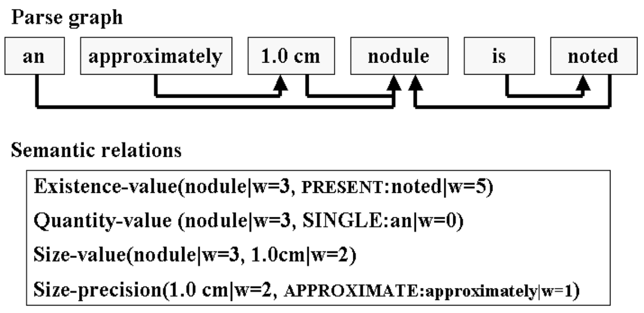
1. Structural analyzer
2. Lexical analyzer
3. Syntactic parser
4. Semantic interpreter
5. Frame builder
6. Coreference resolution

The structural analyzer uses a maximum entropy classifier [Ratnaparkhi, 1998] to determine the sentence boundaries after dividing the document into sections based on page layout cues. The lexical analyzer looks up each word of the sentence in a domain-specific glossary. This assigns syntactic and semantic features to each word that are used later by the parser and the semantic interpreter. A glossary with about 5,000 entries was manually constructed for the domain of thoracic radiology reports. In addition to glossary look-up, the lexical analyzer uses rules to interpret dates and numerical expressions such as “1.2 x 1.7 cm”.

The syntactic parser creates a dependency graph with an arc from each word to the word it modifies. We use a statistical parser somewhat similar to that of Collins [1996] and of Eisner [1996], in which the probability of an arc between each pair of words is estimated from hand-tagged training sentences. Dynamic programming is used to compute the parse graph with maximum joint probability, subject to some constraints on well formedness of the graph.

The next step of processing, semantic interpretation, is described in detail later. It learns a set of semantic rules from hand-tagged training examples using a covering algorithm [Quinlan, 1990; Clark and Niblett, 1989]. Each rule traverses nodes in the parse graph, testing semantic and syntactic features. When a rule succeeds, it creates a semantic relation for a pair of phrases in the sentence.

Figure 2 shows an example of a dependency graph for the sentence “An approximately 1.0 cm nodule is noted” and the semantic relations derived from the sentence. The arc from “noted” (word 5) to “nodule” (word 3) is interpreted as an *Existence-value* relation with the value of PRESENT. The arc from “1.0 cm” to “nodule” is a *Size-value* relation, while the arc from “approximately” to “1.0 cm” is a *Size-precision* relation with the value of APPROXIMATE.



Existence-value(nodule|w=3, PRESENT:noted|w=5)
Quantity-value (nodule|w=3, SINGLE:an|w=0)
Size-value(nodule|w=3, 1.0cm|w=2)
Size-precision(1.0 cm|w=2, APPROXIMATE:approximately|w=1)

Figure 2. Semantic relations derived from a parse graph

The following step is the frame builder that takes a set of semantic relations between pairs of words and assembles them into semantic frames. In the domain of thoracic radiology, the topic of each frame is a medical finding with a list of its properties. Figure 1 shows an example of a semantic frame with *Existence, Quantity, Size, Location,* and *Size trend*.

The last step is coreference resolution that tracks references to a frame topic across sentences. For each finding, the probability is estimated that it co-refers to each previously mentioned finding based on semantic features and lexical cues (e.g. “the nodule” vs. “a nodule”). The current version of the coreference module uses hand-code rules to compute probabilities, although we intend to derive probabilities from training in the future.

An example of coreference resolution is found in Figure 1, where the “mass lesion” in sentence one is equal to “it” in sentences two and three and to “lesion” in sentence four. The semantic frame shown in the figure has been merged from frames for the individual sentence references.

3 Output design: representing meaning

A precise and unambiguous representation of the target concepts for a domain is essential for a text understanding system. The output representation must be expressive enough to capture important details and to handle the full range of relevant information for the domain. In a medical domain it is important to make a distinction between “there is a nodule” and “there may be a nodule” and to note qualifiers such as “fairly round” or “slight increase”.

We chose to represent the output as semantic frames, each with a topic and a list of properties of that topic. The properties may in turn have one or more fields. In the radiology domain, a topic may be a finding such as “nodule”. A property of the nodule may be *Existence* with fields such as *Existence-value* = PRESENT, *Existence-certainty* = POSSIBLE. Another property of the nodule might be *Shape* with *Shape-value* = “round” and *Shape-degree* = MODERATE_DEGREE.

Where the system designer can anticipate possible field values, the output has a symbolic value rather than the actual phrase from the text. The *Existence-certainty* field can be DEFINITE, POSSIBLE, LIKELY, or UNLIKELY. The actual phrase such as “there is” or “may be” is also retained to give evidence for the set value. For some fields the actual phrase from the text serves as the field value.

This three-tiered structure (topic, property, field) seems in practice to be sufficiently expressive to capture nearly all the details about findings that are typically reported by radiologists. A flat list of frame slots would not be sufficient, and a deeply nested structure would add unnecessarily to computational complexity. We currently have 19 properties plus the catchall *Other property*, some of them highly specific to lung masses such as *Calcification, Homogeneity, External architecture,* and *Internal architecture*.

Since the intent is a domain-independent system with domain-specific knowledge, none of the property names or field names is hard coded. The system developer provides a declarative frame definition that lists properties and fields, and may also create new properties or fields on the fly while tagging new training examples.

4 Designing a semantic rule learner

As an intermediate step towards creating the semantic frames, the system creates local fragments of meaning that we call *semantic relations*. A semantic relation gives a domain-specific label to the relation between pairs of phrases in a sentence.

Semantic Relation has:
- Relation name
- Head phrase
- Modifier phrase
- Set value

A semantic relation has a name, such as *Existence-value*, that is composed of the frame property name and field name. Tying the relation name to the frame properties allows the system to build frames automatically from a set of semantic relations with no domain-specific code.

The semantic relation is grounded firmly in the actual text by two arguments, the head phrase and the modifier phrase, each of which is represented as a pair of indices into the sentence for the starting and ending word of the phrase. An optional set value gives one of a predefined set of symbolic values to the semantic relation. Figure 3 shows an example of semantic relations involving location of a finding. The relation between "mass" (word 1) and the "right upper lobe" (word 4 to 6) is *Location-value*. The right upper lobe is further modified by "within" (word 2) to give a *Location-relation* with set value = IN.

Sentence:
"The mass within the upper right lobe …"

Semantic relations:
Location-value (mass|w=1, upper right lobe|w=4|w=6)
Location-relation (upper right lobe|w=4|w=6, IN:within|w=2)

Figure 3: two semantic relations derived from a sentence

The tagging of training examples for semantic rule learning is done at the level of semantic frames rather than individual semantic relations. There are two reasons for this: semantic relations are an internal representation that would be less intuitive for an end user, and it is vital for the semantic relations to be tightly connected with the frames they are to produce.

Each field value in the training set must be grounded in one or more reference from the text, with the exception of default values, such as *Existence-value* = PRESENT. The system derives training examples for semantic relations automatically from the hand-tagged semantic frames. If more than one reference phrase is given for a field value, multiple semantic relations are created. For example, if *Existence-value* = PRESENT has reference of "there is" and also "noted", two semantic relations are created, one with each of the references as the modifier phrase.

For the experiments reported here, it took about 12 hours to tag the first 30 sentences. The system derived a provisional set of rules that was used to assign tags automatically to the next 300 sentences, which took another 12 hours to edit by hand.

A wide number of machine learning algorithms could be used to create semantic relations from a set of training examples. Rule learning seems more appropriate for this task than a machine learning classifier. Since there are O(n^4) possible pairs of phrases to consider in each sentence, a classifier needs a preliminary step that decides on candidate pairs of phrases before deciding whether each candidate constitutes a given semantic relation.

The semantic interpreter takes as its input, the parse graph created by the syntactic parser and the syntactic and semantic tags assigned to individual words by the lexical analyzer. Semantic rules traverse the parse graph, testing constraints on syntactic and semantic features. Figure 4 shows a training example from a sentence such as "a … mass … is noted" that has been tagged as *Existence-value* = PRESENT. Two possible semantic rules are shown that operate correctly on this example. The first rule looks in the parse graph for a word with syntactic features *pastp* and semantic features *exist.observed.noted*. If this is found, follow a forward arc (>) to a word to the left in the sentence (L) with syntactic features *noun.sing* and semantic features *abnorm.physobj.finding.lesion*.

Semantic training (hand-tagged)

```
Topic: mass   [word=3]
  Existence   value: PRESENT: noted  [word=10]
```

Rule with most specific features for this example

Existence-value	PRESENT	
. . pastp	exist.observed.noted	mod
L > noun.sing	abnorm.physobj.finding.lesion	head

Most general rule that fits all training examples

Existence-value	PRESENT	
. . any	exist.observed	mod
. > any	abnorm	head

Figure 4. Two possible semantic rules from a training example

This first rule requires all features of the words and is too specific to be broadly applicable. The second rule also fits this example, but has dropped many of the constraints. Dropping further constraints would cause the rule to apply to examples that had not been tagged as *Existence-value* = PRESENT. The goal of the rule learner is to find the most general rule that operates properly on the entire training set.

This type of rule learner is called a covering algorithm and resembles FOIL [Quinlan, 1990], CN2 [Clark and Niblett, 1989], and CRYSTAL [Soderland et al., 1995; Soderland, 1997]. A training example is selected as a seed. The rule learner searches for the most general rule that matches the seed example, testing each proposed version of the rule against the entire training set. When a rule is accepted, all examples covered by the rule are marked as *covered*. The rule learner continues to select seed examples and derive rules until no uncovered examples remain. The algorithm is outlined in Figure 5.

```
RULE_LEARNER (Training_set) {
  Do until all Training_set is covered {
    Select Seed from Training_set
    Rule = GENERATE_RULE (Seed, Training_set)
    Save Rule
    Mark examples in Training_set covered by Rule
  }
}

GENERATE_RULE (Seed, Training_set) {
  Rule = most general rule for Seed
  Initialize Beamset with Rule
  Test Rule on Training_set
  If Rule is within error_tolerance {
    return Rule
  } else {
    Do until Beamset is empty or rule is found {
      Beamset = k best specializations of
                all rules in Beamset
      Rule = best rule in Beamset
      Test Rule on Training_set
      If Rule is within error_tolerance {
        return Rule
      }
    }
  }
}
```

Figure 5: Rule learning algorithm

In searching for the most general rule, the algorithm starts with a general initial rule that has no syntactic constraints and only the most general semantic constraint for each word. A beam search is used to specialize this initial rule, where each specialization adds one constraint to a rule in the beamset. The best k of those specializations are kept in the beamset, where the goodness metric is Laplacian expected error. A beam size of k=5 was used in experiments reported here. The search takes time $O(n^2kr)$ where n is the number of training examples, k is the beam size, and r is the number of features in a rule.

To accommodate noise in the data, the search halts when a rule is found with error rate within an error tolerance. This is an overly simple means of handling noisy data. A more principled approach, although more computationally expensive, is to hold out a separate tuning set. First learn a set of rules on the training set with zero error tolerance, then test the effect on the tuning set of generalizing or eliminating individual rules.

One challenge that arose in designing a fully automated semantic interpreter is the problem of conflicting rules. Figure 6 gives an example of how two rules can apply to the same sentence that give contradictory semantic relations. Rule 2 says that when a word similar to "seen" modifies a word similar to "nodule", the Existence-*value* is PRESENT. Rule 1 looks for a word such as "not" that modifies "seen" and assigns the *Existence-value* of ABSENT. Unfortunately, both rules apply to "nodule was not seen".

One solution would be domain-specific code that handles such conflicts, but this would violate the goal of a domain-independent system with machine learning for the domain knowledge. An even more pernicious problem is that the rule learner will be unable to learn Rule 2 – this rule operates incorrectly on examples such as "nodule was not seen" and appears to have an error rate of about 30%.

Our solution is a data file that lists set values for each field in order of precedence, which is provided as part of the target output definition. The system will ignore a semantic rule that gives a lower precedence value in the case of a conflict. The rule learner is able to learn Rule 2 of Figure 6 if it learns rules for semantic relations in order of precedence. By the time it is learning *Existence-value* PRESENT it has already learned rules for *Existence-value* ABSENT, so the output of Rule 2 on "nodule was not seen" will be ignored and not counted as an error.

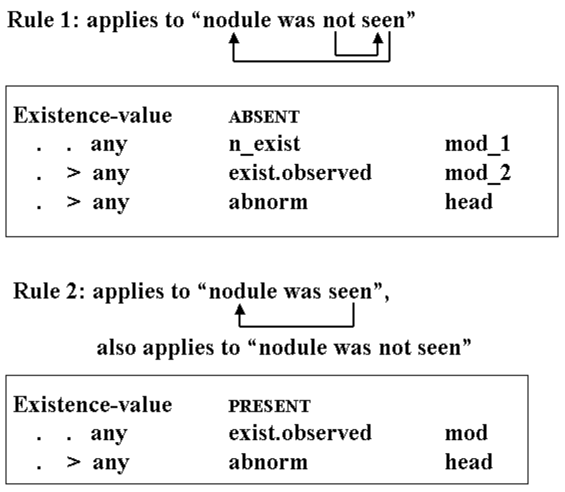

Rule 1: applies to "nodule was not seen"
Existence-value ABSENT
. . any n_exist mod_1
. > any exist.observed mod_2
. > any abnorm head
Rule 2: applies to "nodule was seen",
also applies to "nodule was not seen"
Existence-value PRESENT
. . any exist.observed mod
. > any abnorm head

Figure 6. An example of conflicting rules that both apply to "nodule was not seen" with conflicting results

5 Empirical Results

Experiments were conducted on a set of chest radiology reports of cancer patients. A training set of 45 reports comprising 329 sentences was hand-tagged with parser training and semantic interpreter training. The full set was divided into ten equal partitions, and sentences from each partition were tested using parser statistics and semantic rules learned from reports in the other nine partitions.

The target concept being learned was a semantic frame for each clinical finding in the report, such as "mass", "nodule", "atelectasis", and more general references that could refer to those findings such as "density", "finding", "it", or "this". The semantic frames list all properties of the finding mentioned in the report, as well as phrases that modify or qualify those properties. The 329 training sentences produced about 2400 semantic relations.

Evaluation of the semantic interpreter was done by comparing the semantic relations produced by the system with an answer key of those derived from the hand-tagged training. The evaluation metrics are *recall* and *precision*, where recall is the number of correct responses divided by the possible responses and precision is the number correct divided by the total responses. If a semantic relation reported by the system matched one in the answer key, the response was counted as correct, otherwise as an error. A response may partially match the answer key if the reference phrases have words in common but are not identical (Location value of "lung base" instead of "right lung base"). This is counted as half correct, half error. In the experiment reported here, giving partial credit makes a difference of a little less than two recall and precision points over calling such cases an error.

Semantic relations were merged if they had the same set value and differed only in the modifier phrase that gave evidence for that set value. For example, consider two *Existence-value* relations that have the same head phrase and the same set value PRESENT, but one has modifier phrase "there is" and the other has modifier phrase "seen". These are counted as a single semantic relation. If a semantic relation from the system matches one in the answer key with the same set value, but a different modifier phrase, this is counted as correct.

Table 1 shows results of the semantic interpreter at various parameter settings. The error tolerance parameter allows the rule learner to create the most general rule with error rate less than a threshold. A minimum coverage parameter higher than 1 causes the system to ignore rules that apply to fewer than that number of training examples.

Table 1: Results of semantic interpreter for 10-fold cross validation at various parameter settings

Parameter settings		All sentences		No conjunctions	
Err tol	Min cover	R	P	R	P
0.10	1	73.7	83.2	78.0	83.3
0.10	2	69.7	85.8	74.2	86.5
0.10	5	63.1	86.5	67.4	87.1
0.05	1	73.5	84.4	77.9	85.3
0.05	2	69.5	87.2	74.1	88.8
0.05	5	62.8	88.1	67.3	89.8
0.00	1	72.9	84.5	77.1	85.4
0.00	2	68.5	87.6	72.9	89.0
0.00	5	60.4	88.7	64.9	90.2

The best balanced recall and precision is recall 73.5 and precision 84.4 at an error tolerance of 0.05. The error tolerance parameter has a smaller impact than expected because of the relatively small training set. Only a small fraction of the rules covered more than five training examples and more than half of the rules covered just a single training example. Even at error tolerance 0.10 a rule that covers nine examples must make no errors.

Setting the minimum coverage parameter greater than 1 eliminates the least reliable, low coverage rules, most of which would not apply to a test sentence anyway. This has the effect of raising precision with a corresponding drop in recall. Highest precision comes from setting the error tolerance to 0.0 and discarding all but the highest coverage rules. In this domain, dropping some of the low coverage rules actually decreases precision, as rules are dropped that distinguish between "seen" and "no longer seen" and other exceptions to more general rules.

The parser makes more errors on sentences containing conjunctions than on other sentences, with parser recall 82.0 at precision 81.9 on sentences with conjunctions and recall 88.3 at precision 88.3 on sentences without conjunctions. The column on the right in Table 1 shows semantic interpreter results on sentences without conjunctions, where recall is generally about four points higher than on all sentences.

It is hard to make a meaningful comparison with other text understanding systems, since domains and difficulty of the tasks vary so much. In the seminar announcement domain, Freitag's system found the correct speaker 62% of the time and the correct starting time 99% of the time [Freitag, 1998]. Califf and Mooney [1997] reported recall of 53 at precision 84 in the job posting domain. Better performance was reported by subsequent systems that were tested on these data sets. Freitag and McCallum [2000] had F measure (combined recall and precision) of 76.9 for the speaker slot of seminar announcements. Ciravegna [2000] reported recall 70 at precision 87 for seminar speaker and recall 82 at precision 86 for the job posting data set.

The best systems from the MUC conferences had recall and precision between 50 and 60 for the full template creation task. Scores were higher for the simpler TR (template relation) task in later MUC conferences; BBN [Miller et al. 1998] got TR recall 64 at precision 81. The results shown in Table 1 are for semantic relations between words or phrases within sentences, before they have been combined into templates or merged across sentences. Results can be expected to degrade somewhat after merging of semantic output based on imperfect coreference resolution.

6 Conclusions

While there is no general-purpose text understanding system, considerable progress has been made in recent years with systems that focus on extracting information from narrowly defined domains. To adapt such a system to a new domain, a system developer must create a precise definition of the target concepts for the domain. How close the system comes to in-depth understanding is limited primarily by the coverage and level of detail in the output representation. In a domain such as radiology reports, the representation can be expressive enough to capture nearly all details about properties of the clinical findings reported.

A goal for text understanding systems is to use domain-independent code and to rely on machine learning and statistical techniques for the bulk of the domain knowledge.

Creating hand-tagged training examples is arguably less labor intensive than creating complex rule sets manually, although still not a trivial labor cost. We have presented a system that derives parser statistics and semantic interpreter rules from training examples, with a probabilistic coreference module that could also be adapted to machine learning. Performance is at least on a par with systems that use extensive hand-coded rules.

One area of hand-coded knowledge could not be avoided: a glossary with syntactic and semantic features of individual words. Work is needed to streamline glossary creation as much as possible. In the radiology domain, a small amount of domain-specific code also remains that replaces anatomic phrases with the preferred term from a medical controlled vocabulary and also code that reformats size expressions. Other than that, no domain-specific code is used to assemble semantic frames from the automatically learned semantic relations.

Acknowledgements

This work was supported by grant PO1 CA51198 from the National Cancer Institute. The system was developed jointly with Ricky Taira, Rex Jakobovits, and Ashley Warner.

References

[Califf and Mooney, 1997] Mary Elaine Califf and Raymond Mooney. Relational learning of pattern-matching rules for information extraction. In *Working Papers of the ACL-97 Workshop on Natural Language Learning*, pages 9-15, 1997.

[Ciravegna, 2000] Fabio Ciravegna. Learning to tag information extraction from text. In *ECAI-200 workshop on Machine Learning for Information Extraction*, 2000.

[Clark and Niblett, 1989] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning* 3:261-283, 1989.

[Collins, 1996] Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184-191, 1996.

[Eisner, 1996] Jason Eisner. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of COLING-96*, pages 340-345, 1996.

[Freitag, 1998] Dayne Freitag. Multistrategy learning for information extraction. In *Proceedings of the Fifteenth International Machine Learning Conference*, pages 161-169, 1998.

[Freitag and Kushmerick, 2000] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *ECAI-200 workshop on Machine Learning for Information Extraction*, 2000.

[Freitag and McCallum, 2000] Dayne Freitag and Andrew McCallum. Information extraction with HMM structures learned by stochastic optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000). 2000.*

[Friedman et al., 1994] C. Friedman, P. Alderson, J. Austin, et al. A general natural language text processor for clinical radiology. *Journal of American Medical Informatics Association (JAMIA)*, 1(2):161-174, 1994.

[Friedman, 1997] Carol Friedman. Towards a comprehensive medical language processing system: methods and issues. In *AMIA*, pages 595-599, 1997.

[Lehnert et al., 1993] Wendy Lehnert et al. UMass/ Hughes: Description of the CIRCUS system used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference*, Morgan Kaufmann, San Francisco, pages 277-292, 1993.

[Miller et al., 1998] Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. BBN: Description of the SIFT system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference*, Morgan Kaufmann, San Francisco, 1998.

[MUC-4, 1992] *Proceedings of the Fourth Message Understanding Conference*, Morgan Kaufmann, San Francisco, 1992.

[MUC-5, 1993] *Proceedings of the Fifth Message Understanding Conference*, Morgan Kaufmann, San Francisco, 1993.

[Quinlan, 1990] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3): 239-266.

[Ratnaparkhi, 1998] Adwait Ratnaparkhi. Maximum entropy models for natural language ambiguity resolution. PhD. dissertation, Dept. of Computer and Information Science, University of Pennsylvania, 1998.

[Soderland et al., 1995] Stephen Soderland, David Fisher, Jon Aseltine, and Wendy Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314-1321, 1995.

[Soderland, 1997] Stephen Soderland. Learning text analysis rules for domain-specific natural language processing. PhD. dissertation, University of Massachusetts, Amherst, 1997.

[Soderland, 1999] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34: 233-272, 1999.

[Taira and Soderland, 2001] Ricky Taira, Stephen Soderland, and Rex Jakobovits. Automatic structuring of radiology free text reports. *Radiographics*, 21, 2001.