Corpus Based Statistical Generalization Tree in Rule Optimization *

Joyce Yue Chai Alan W. Biermann
Department of Computer Science
Box 90129, Duke University
Durham, NC 27708-0129
chai@cs.duke.edu awb@cs.duke.edu

Abstract

A corpus-based statistical Generalization Tree model is described to achieve rule optimization for the information extraction task. First, the user creates specific rules for the target information from the sample articles through a training interface. Second, WordNet is applied to generalize noun entities in the specific rules. The degree of generalization is adjusted to fit the user's needs by use of the statistical Generalization Tree model. Finally, the optimally generalized rules are applied to scan new information. The results of experiments demonstrate the applicability of our Generalization Tree method.

Introduction

Research on corpus-based natural language learning and processing is rapidly accelerating following the introduction of large on-line corpora, faster computers, and cheap storage devices. Recent work involves novel ways to employ annotated corpus in part of speech tagging (Church 1988) (Derose 1988) and the application of mutual information statistics on the corpora to uncover lexical information (Church 1989). The goal of the research is the construction of robust and portable natural language processing systems.

The wide range of topics available on the Internet calls for an easily adaptable information extraction system for different domains. Adapting an extraction system to a new domain is a tedious process. In the traditional customization process, the given corpus must be studied carefully in order to get all the possible ways to express target information. Many research groups are implementing the efficient customization of information extraction systems, such as BBN (Weischedel 1995), NYU (Grishman 1995), SRI (Appelt, Hobbs, et al 1995), SRA (Krupka 1995), MITRE (Aberdeen, Burger, et al 1995), and UMass (Fisher, Soderland, et al 1995).

*This work has been supported by a Fellowship from IBM Corporation.

We employ a rule optimization approach and implement it in our trainable information extraction system. The system allows the user to train on a small amount of data in the domain and creates the specific rules. Then it automatically extracts a generalization from the training corpus and makes the rule general for the new information, depending on the user's needs. In this way, rule generalization makes the customization for a new domain easier.

This paper specifically describes the automated rule optimization method and the usage of WordNet (Miller 1990). A Generalization Tree (GT) model based on the training corpus and WordNet is presented, as well as how the GT model is used by our system to automatically learn and control the degree of generalization according to the user's needs.

System Overview

The system contains three major subsystems which, respectively, address training, rule optimization, and the scanning of new information. The overall structure of the system is shown in Figure 1. First, each article is partially parsed and segmented into Noun Phrases, Verb Phrases and Prepositional Phrases. An IBM LanguageWare English Dictionary and Computing Term Dictionary, a Partial Parser [1], a Tokenizer and a Preprocessor are used in the parsing process. The Tokenizer and the Preprocessor are designed to identify some special categories such as e-mail address, phone number, state and city etc. In the training process, the user, with the help of a graphical user interface(GUI) scans a parsed sample article and indicates a series of semantic net nodes and transitions that he or she would like to create to represent the information of interest. Specifically, the user designates those noun phrases in the article that are of interest and uses the interface commands to translate them

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

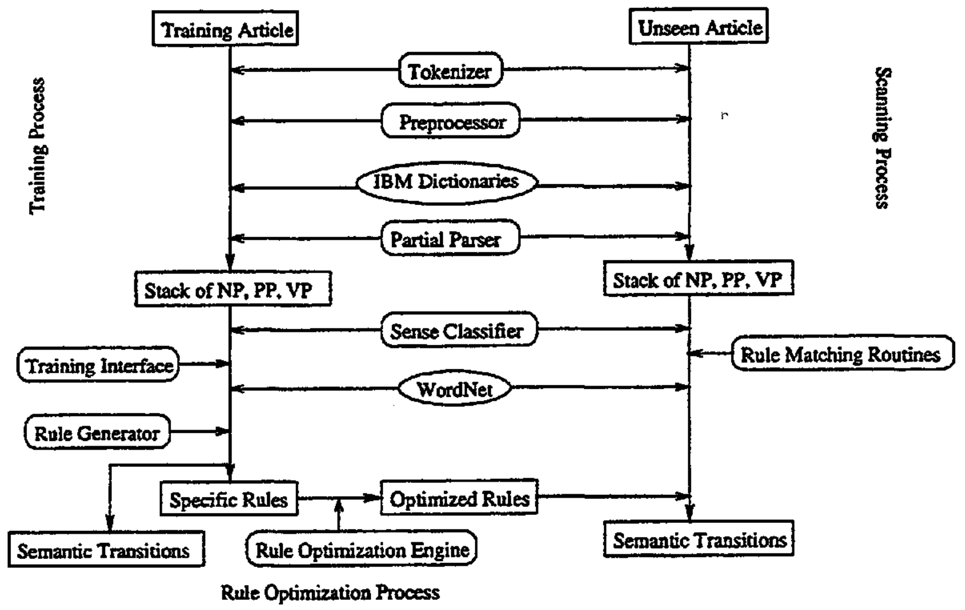


Figure 1: System Overview

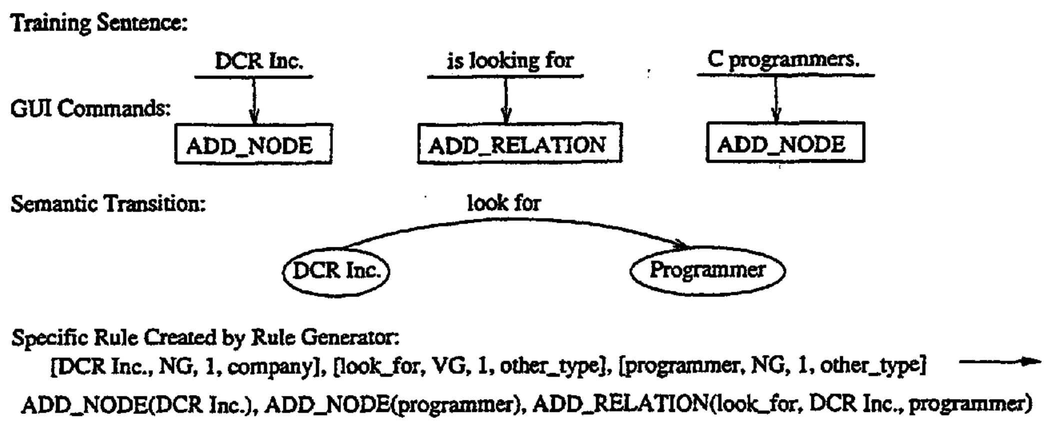
into semantic net nodes. Furthermore, the user designates verb phrases and prepositions that relate the noun phrases and uses commands to translate them into semantic net transitions between nodes. In the process, the user indicates the desired translation of the specific information of interest into semantic net form that can easily be processed by the machine. For each headword in a noun phrase, WordNet is used to provide sense information. Usually 90% of words in the domain are used in sense one (the most frequently used sense) as defined in WordNet. However, some words might use other sense. For example, "opening" often appears in the job advertisement domain. But instead of using the first sense as {*opening, gap*}, it uses the fourth sense as {*opportunity, chance*}. Based on this scenario, for headwords with senses other than sense one, the user needs to identify the appropriate senses, and the Sense Classifier will keep the record of these headwords and their most frequently used senses. When the user takes the action to create the semantic transitions, a Rule Generator keeps track of the user's moves and creates the rules automatically. These rules are specific to the training articles and they need to be generalized in order to be applied on other unseen articles in the domain. According to different requirements from the user, the Rule Optimization Engine, based on WordNet, generalizes the specific rules created in the training process and forms a set of optimized rules for processing new information. This rule optimization process will be explained in the later sections. During the scanning of new information, with the help of a rule matching routine, the system applies the optimized rules on a large number of unseen articles from the domain. For the most headwords in the phrases, if they are not in the Sense Classifier table, sense one in WordNet will be assigned; otherwise, the Sense Classifier will provide the system their most frequently used senses in the domain. The output of the system is a set of semantic transitions for each article that specifically extract information of interest to the user. Those transitions can then be used by a Postprocessor to fill templates, answer queries, or generate abstracts (Bagga, Chai 1997).

Rule Operations

Our trainable information extraction system is a rule-based system, which involves three aspects of rule operations: rule creation, rule generalization and rule application.

Rule Creation

In a typical information extraction task, the most interesting part is the events and relationships holding among the events (Appelt, Hobbs, et al 1995). These relationships are usually specified by verbs and prepositions. Based on this observation, the left hand side (LHS) of our meaning extraction rules is made up of three entities. The first and the third entities are the target objects in the form of noun phrases, the second entity is the verb or prepositional phrase indicating the relationship between the two objects. The right hand side (RHS) of the rule consists of the operations



```
[DCR Inc., NG, 1, company], [look_for, VG, 1, other_type], [programmer, NG, 1, other_type] ——→
ADD_NODE(DCR Inc.), ADD_NODE(programmer), ADD_RELATION(look_for, DCR Inc., programmer)
```

Figure 2: The Rule Creation Process

required to create a semantic transition–ADD_NODE, ADD_RELATION.

For example, during the training process, as shown in Figure 2, the user trains on the sentence "DCR Inc. is looking for C programmers...", and would like to designate the noun phrases(as found by the parser) to be semantic net nodes and the verb phrase to represent a transition between them. The training interface provides the user ADD_NODE, ADD_RELATION GUI commands to accomplish this. ADD_NODE is to add an object in the semantic transition. ADD_RELATION is to add a relationship between two objects. The specific rule is created automatically by the rule generator according to the user's moves.

Rule Generalization

The rule created by the rule generator as shown in Figure 2 is very specific, and can only be activated by the training sentence. It will not be activated by other sentences such as "IBM Corporation seeks job candidates in Louisville...". Semantically speaking, these two sentences are very much alike. Both of them express that a company is looking for professional people. However, without generalization, the second sentence will not be processed. So the use of the specific rule is very limited. 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 use the power of WordNet to achieve generalization.

Introduction to WordNet WordNet is a large-scale on-line dictionary developed by George Miller and colleagues at Princeton University (Miller, et al 1990a). The most useful feature of WordNet to the Natural Language Processing community is its attempt to organize lexical information in terms of word meanings, rather than word forms. Each entry in WordNet is a concept represented by the synset. A synset is a list of synonyms, such as {*engineer, applied scientist, technologist*} . The information is encoded in the form of semantic networks. For instance, in the network for nouns, there are "part of", "is_a", "member of".... relationships between concepts. Philip Resnik wrote that "...it is difficult to ground taxonomic representations such as WordNet in precise formal terms, the use of the WordNet taxonomy makes reasonably clear the nature of the relationships being represented..." (Resnik 1993). The hierarchical organization of WordNet by word meanings (Miller 1990) provides the opportunity for automated generalization. With the large amount of information in semantic classification and taxonomy provided in WordNet, many ways of incorporating WordNet semantic features with generalization are foreseeable. At this stage, we only concentrate on the Hypernym/Hyponym feature.

A hyponym is defined in (Miller, et al 1990a) as follows: " A noun X is said to be a hyponym of a noun Y if we can say that *X is a kind of Y*. This relation generates a hierarchical tree structure, i.e., a taxonomy. A hyponym anywhere in the hierarchy can be said to be "a kind of" all of its superordinateds. ..." If X is a hyponym of Y, then Y is a hypernym of X.

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

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

1. **An Abstract Specific 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_2), ADD_RELATION(w_2, w_1, w_3)

2. **A Generalized Rule:**

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

Figure 3: Sample Rules

sp = (programmer, NG, 1, other_type)

$\Big\downarrow$ various generalization degree

Generalize(sp, 1) = {engineer, applied scientist, technologist}
Generalize(sp, 2) = {person, individual, someone,...}
Generalize(sp, 3) = {life form, organism, being, ...}
Generalize(sp, 4) = {entity}

Figure 4: Generalization for a Specific Concept

way of locating the superordinate concepts of *sp*. By following additional Hypernymy, we will get more and more generalized concepts and eventually reach the most general concept, such as {*entity*}. As a result, 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 (Chai, Biermann 1997).The function to accomplish this task is *Generalize(sp,h)*, which returns a hypernym *h* levels above the concept *sp* in the hierarchy. *Generalize(sp,0)* returns the synset of *sp*. For example, in Figure 4, the concept {*programmer*} is generalized at various levels based on Wordnet Hierarchy. WordNet is an acyclic structure, which suggests that a synset might have more than one hypernym. However, this situation doesn't happen often. In case it happens, the system selects the first hypernym path.

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 hierarchy in WordNet by performing the $Generalize(sp, h)$ function. The degree of generalization for rules varies with the variation of h in $Generalize(sp, h)$.

Figure 3 shows an abstract generalized rule. The $\in$ symbol signifies the subsumption relationship. Therefore, $a \in b$ signifies that a is subsumed by b, or concept b is a superordinate concept of concept a.

Optimization Rules with different degrees of generalization on their different constituents will have a different behavior when processing new texts. A set of generalized rules for one domain might be sufficient; but in another domain, they might not be. Within a particular rule, the user might expect one entity to be relatively specific and the other entity to be more general. For example, if a user is interested in finding all DCR Inc. related jobs, he/she might want to hold the first entity as specific as that in Figure 2, and generalize the third entity. The rule optimization process is to automatically control the degree of generalization in the generalized rules to meet user's different needs. Optimization will be described in later sections.

Rule Application

The optimally generalized rules are applied to unseen articles to achieve information extraction in the form of semantic transitions. The generalized rule states that the RHS of the rule gets executed if *all* of the following conditions satisfy:

- 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.

Figure 5 shows an example of rule matching and creating a semantic transition for the new information. In

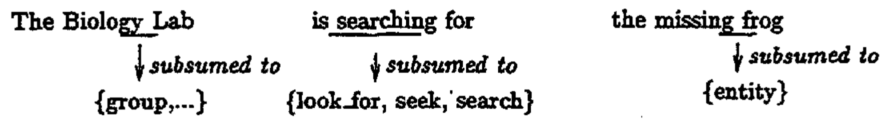
Specific Rule:

[DCR Inc., NG, 1, company], [look_for, VG, 1, other_type], [programmer, NG, 1, other_type]
$\longrightarrow$ ADD_NODE(DCR Inc.), ADD_NODE(programmer), ADD_RELATION(look_for, DCR Inc., programmer)

Generalized to the Most General Rule:

$(W_1, C_1, S_1, T_1) \in \{group, ...\}, (W_2, C_2, S_2, T_2) \in \{look\_for, seek, search\}, (W_3, C_3, S_3, T_3) \in \{entity\}$
$\longrightarrow$ ADD_NODE(W_1, ADD_NODE(W_3), ADD_RELATION(W_2, W_1, W_3)

Unseen Sentence:



Execute the RHS of the Rule:

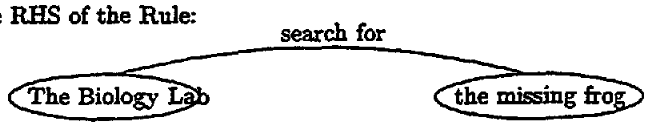


Figure 5: The Application of Generalized Rules

the example, the most general rule is created by generalizing the first and the third entities in the specific rule to their top hypernyms in the hierarchy. Since verbs usually have only one level in the hierarchy, they are generalized to the synset at the same level.

Rule Optimization

The specific rule can be generalized to the most general rule as in Figure 5. When we apply this most general rule again to the training corpus, a set of semantic transitions are created. Some transitions are relevant, while the others are not. Users are expected to select the relevant transitions through a user interface. We need a mechanism to determine the level of generalization that can achieve best in extracting the relevant information and ignoring the irrelevant information. Therefore, Generalization Tree (GT) is designed to accomplish this task. While maintaining the semantic relationship of objects as in WordNet, GTs collect the relevancy information of all activating objects and automatically find the optimal level of generalization to fit the user's needs. A database is used to maintain the relevancy information for all the objects which activate each most general concept in the most general rule. This database is transformed to the GT structure, which keeps the statistical information of relevancy for each activating object and the semantic relations between the objects from WordNet. The system automatically adjusts the generalization degrees for each noun entity in the rules to match the desires of the user. The idea of this optimization process is to first keep recall as high as possible by applying the most general rules, then adjust the precision by tuning the rules based on the user's specific inputs.

An Example of GT

Suppose we apply the most general rule in Figure 5 to the training corpus, and the entity three in the rule is activated by a set of objects shown in Table 1. From a user interface and a statistical classifier, the relevancy_rate(*rel_rate*) for each object can be calculated.

$$rel\_rate(obj) = \frac{count\ of\ obj\ being\ relevant}{total\ count\ of\ occurence\ of\ obj}$$

As shown in Table 1, for example, $rel\_rate(\{analyst...\}) = 80\%$, which indicates that when $\{entity\}$ in the most general rule is activated by *analyst*, 80% of time it hits relevant information and 20% of time it hits irrelevant information. On the other hand, it suggests that if $\{entity\}$ is replaced by the concept $\{analyst...\}$, a roughly 80% precision could be achieved in extracting the relevant information. The corresponding GT for Table 1 is shown in Figure 6.

In GT, each activating object is the leaf node in the tree, with an edge to its immediate hypernym (parent). For each hypernym list in the database, there is a corresponding path in GT. Besides the ordinary hard edge represented by the solid line, there is a soft edge represented by the dotted line. Concepts connected by the soft edge are the same concepts. The only difference between them is that the leaf node is the actual activating object, while the internal node is the hypernym for other activating objects. Hard edges and soft edges only have representational difference, as to the calculation, they are treated the same. Each

object	sense	hypernym list	depth	rel_rate	count
analyst	1	$\{analyst\} \Rightarrow \{expert\} \Rightarrow \{individual\} \Rightarrow \{life\ form\} \Rightarrow \{entity\}$	4	80%	5
candidate	2	$\{candidate\} \Rightarrow \{applicant\} \Rightarrow \{individual\} \Rightarrow \{life\ form\} \Rightarrow \{entity\}$	4	100%	3
individual	1	$\{individual\} \Rightarrow \{life\ form\} \Rightarrow \{entity\}$	2	100%	5
participant	1	....	5	0%	1
professional	1	...	4	100%	2
software	1	...	5	0%	1
...	...	...	...	...	...

Table 1: Sample Database for Objects Activating {*entity*}

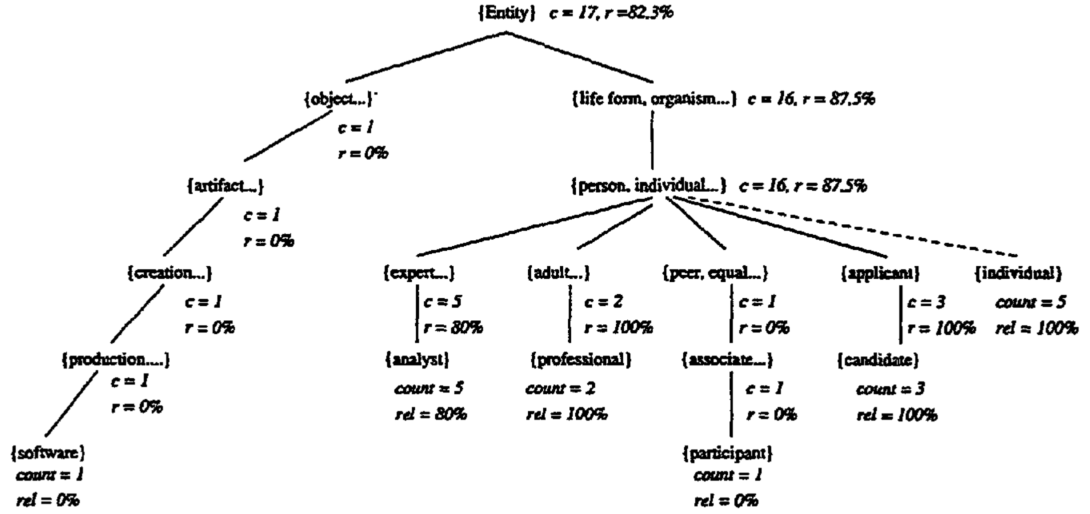


Figure 6: An Example of Generalization Tree

node has two other fields *counts of occurrence* and *relevancy_rate*. For the leaf nodes, those fields can be filled from the database directly. For internal nodes, the instantiation of these fields depends on its children (hyponym) nodes. The calculation will be described later.

If the relevancy_rate for the root node {*entity*} is 82.3%, it indicates that, with the probability 82.3%, objects which activate {*entity*} are relevant. If the user is satisfied with this rate, then it's not necessary to alter the most general concept in the rule. If the user feels the estimated precision is too low, the system will go down the tree, and check the relevancy_rate in the next level. For example, if 87.5% is good enough, then the concept {*life form, organism...*} will substitute {*entity*} in the most general rule. If the precision is still too low, the system will go down the tree, find {*adult..*}, {*applicant..*}, and replace the concept {*entity*} in the most general rule with the union of these two concepts.

Generalization Tree Model

Let's suppose x_n is a noun entity in the most general rule, and x_n is activated by q concepts $e_1^0, e_2^0, ...e_q^0$; the times of activation for each e_i^0 are represented by c_i. Since $e_i^0 (i \leq q)$ activates x_n, there exists a hypernym list $e_i^0 \Rightarrow e_i^1 \Rightarrow .... \Rightarrow x_n$ in WordNet, where e_i^j is the immediate hypernym of e_i^{j-1}. The system maintains a database of activation information as shown in Table 2, and transforms the database to a GT model automatically.

GT is an n-ary branching tree structure with the following properties:

- Each node represents a concept, and each edge represents the hypernym relationship between the concepts. If e_i is the immediate hypernym of e_j, then there is an edge between node e_i and e_j. e_i is one level above e_j in the tree.

- The root node x_n is the most general concept from the most general rule.

activating objects	sense	counts	hypernym list	depth	relevancy_rate
e_1^0	s_1	c_1	$e_1^0 \Rightarrow e_1^1 \Rightarrow ... \Rightarrow x_n$	d_1	r_1
e_2^0	s_2	c_2	$e_2^0 \Rightarrow e_2^1 \Rightarrow ... \Rightarrow x_n$	d_2	r_2
....	...	...	....	....	...
e_q^0	s_q	c_q	$e_q^0 \Rightarrow e_q^1 \Rightarrow ... \Rightarrow x_n$	d_q	r_q

Table 2: database of activating concepts

- The leaf nodes $e_1^0, e_2^0, ...e_q^0$ are the concepts which activate x_n. The internal nodes are the concepts e_j^i ($i \neq 0$ and $1 \leq j \leq q$) from the hypernym paths for the activating concepts.

- For a leaf node e_i^0:
  $counts(e_i^0) = c_i$
  $relevancy\_rate(e_i^0) = r_i$

- For an internal node e, if it has n hyponyms(ie. children) $e_0, ...e_n$ then:

$$counts(e) = \sum_{i=1}^{n} counts(e_i)$$

$$relevancy\_rate(e) = \sum_{i=1}^{n} P(e_i) * relevancy\_rate(e_i)$$

where
$$P(e_i) = \frac{counts(e_i)}{counts(e)}$$

Optimized Rule

For each noun entity in the most general rule, the system keeps a GT from the training set. Depending on user's different needs, a threshold θ is pre-selected. For each GT, the system will start from the root node, go down the tree, and find all the nodes e_i such that $relevancy\_rate(e_i) \geq \theta$. If a node *relevancy_rate* is higher than θ, its children nodes will be ignored. In this way, the system maintains a set of concepts whose *relevancy_rate* is higher than θ, which is called *Optimized-Concepts*. By substituting x_n in the most general rule with *Optimized-Concepts*, an optimized rule is created to meet the user's needs.

The searching algorithm is basically the breadth-first search as follows:

1. Initialize *Optimal-Concepts* to be empty set. Preselect the threshold θ. If the user wants to get the relevant information and particularly cares about the precision, θ should be set high; if the user wants to extract as much as information possible and does not care about the precision, θ should be set low.

2. Starting from the root node x, perform the *Recursive-Search* algorithm, which is defined as the following:
   Recursive-Search(concept x)

```
{ if (relevancy_rate(x) ≥ θ) {
      put x into Optimal-Concepts set;
      exit;
  }
  else {
      let m denote the number of children nodes of x;
      let x_i denote the child of x (0 < i ≤ m);
      for ( i = 1; i ≤ m; i + +)
          Recursive-Search(x_i);
  };
}
```

Experiment and Discussion

In this section we present and discuss results from an experiment. The experimental domain is *triangle.jobs* USENET newsgroup. We trained our system on 24 articles for the extraction of six facts of interests as follows:

- 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.

- Contact Info. Examples: Fax is 919-660-6519, e-mail address.

The testing set contained 162 articles from the same domain as the system was trained on. Out of 162 articles, 21 articles were unrelated to the domain due to the misplacement made by the person who posted them. Those unrelated articles were about jobs

facts	company	position	experience	location	benefit	contact info
training	62.5%	83.3%	91.7%	66.7%	25.0%	95.8%
testing	63.1%	90.8%	90.8%	62.4%	23.4%	97.9%

Table 3: Percentage of Facts in Training and Testing

wanted, questions answered, ads to web site etc. First, we compared some of the statistics from the training set and testing set. The percentage of representation of each fact in the articles for both training and testing domain is shown in Table 3, which is the number of articles containing each fact out of the total number of articles. The distribution of number of facts presented in each article is shown in Figure 7.

The mean number of facts in each article from the training set is 4.39, the standard deviation is 1.2; the mean number of facts in each article from the testing set is 4.35, the standard deviation is 1. Although these statistics are not strong enough to indicate the training set is absolutely the good training corpora for this information extraction task, it suggests that as far as the facts of interest are concerned, the training set is a reasonable set to be trained and learned.

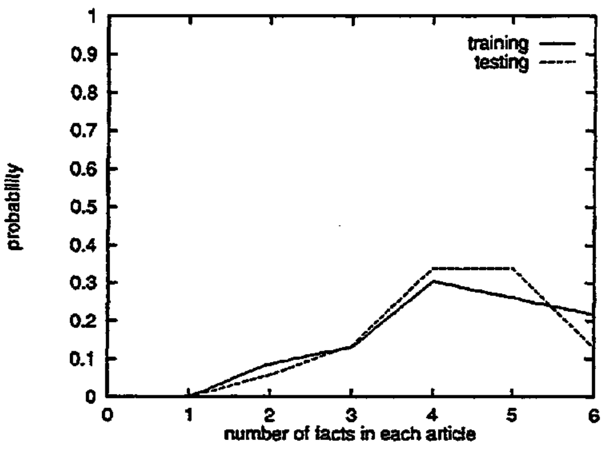


Figure 7: Distribution of Number of Facts in Each Article

The evaluation process consisted of the following steps: first, each unseen article was studied to see if there was any fact of interest presented; second, the semantic transitions produced by the system were examined to see if they correctly extracted the fact of interest. Precision is the number of transitions correctly extracting facts of interest out of the total number of transitions produced by the system; recall is the number of facts which have been correctly extracted out of the total number of facts of interest. The overall performance of recall and precision is defined by the F-measurement (Chinchor 1992), which is

$$\frac{(\beta^2 + 1.0) * P * R}{\beta^2 * P + R}$$

where P is precision, R is recall, $\beta = 1$ if precision and recall are equally important.

First, we tested on single fact extraction, which was *position/title* fact. The purpose of this experiment is to test whether the different θ values will lead to the expected recall, and precision statistics. From the result out of 141 related testing articles, the recall, precision, F-measurement curves are shown in Figure 8. Recall is 51.6% when $\theta = 1.0$, which is lower than 75% at $\theta = 0$, however, precision is the highest at 84.7% when $\theta = 1.0$. The F-measurement achieves its highest value at 64.1% when $\theta = 1.0$.

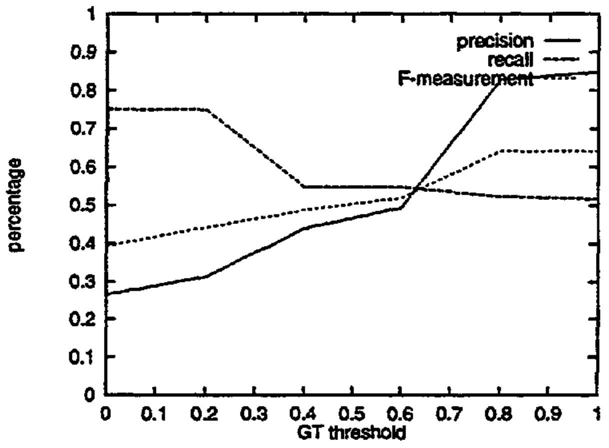


Figure 8: Performance vs. GT Threshold

As mentioned earlier, 21 articles from the testing corpus are unrelated to the job advertisement domain. The interesting question rising here is can we use GT rule optimization method to achieve the information retrieval, in this particular case, to identify those unrelated articles. Certainly, we would hope that optimized rules won't produce any transitions from the unrelated articles. The result is shown in Figure 9. The precision of unrelated articles is the number of articles without any transitions created out of total 21 articles. We can see that, when $\theta = 0.8$, 1.0, precision is 95.7%. Only one article out of 21 articles is mis-identified. But when $\theta = 0$, 0.2, the precision rate is very low, only 28.6%

and 38.1%. If we use the traditional way of keyword matching to do this information retrieval, the precision won't achieve as high as 95.7% since a few resume and job wanted postings will succeed the keyword matching and be mis-identified as related articles.

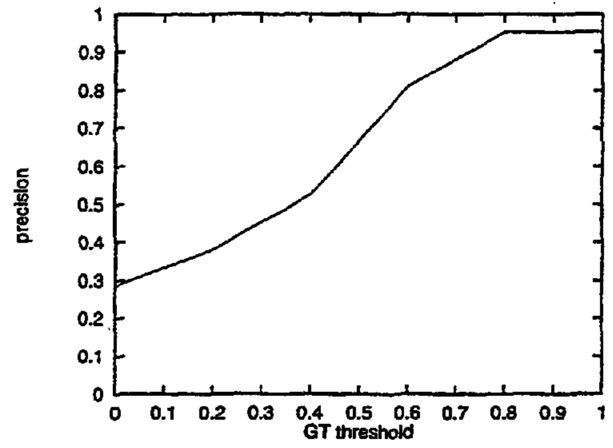


Figure 9: Precision of Identifying Unrelated vs. GT Threshold

The system performance on extracting six facts is shown in Figure 10. The overall performance F-measurement gets to its peak at 70.2% when $\theta = 0.8$. When $\theta = 1.0$, the precision does not get to what we expected. One explanation is that, unlike the extraction of *position/title* fact, for extracting the six facts from the domain, the training data is quite small. It is not sufficient enough to support the user's requirement for a strong estimate of precision. $\theta = 0.8$ is the best choice when the training corpus is small.

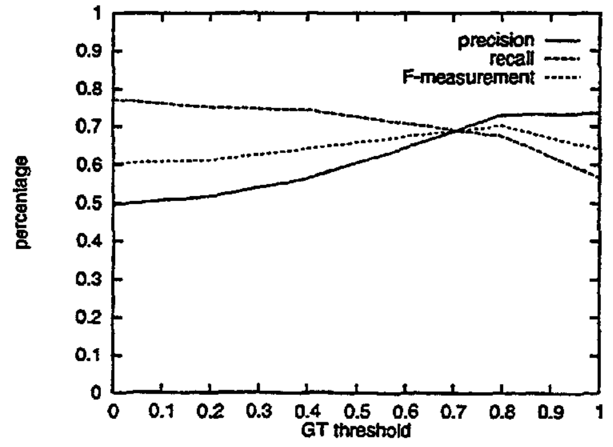


Figure 10: Performance of Extracting Six Facts vs. GT Threshold

Some problems were also detected which prevent better performance of the system. The current domain is a newsgroup, where anyone can post anything which he/she believes is relevant to the newsgroup. It is inevitable that some typographical errors and some abbreviations occur in the articles. And the format of the article sometimes is unpredictable. If we can incorporate into the system a spelling checker, and build a database for the commonly used abbreviations, the system performance is expected to be enhanced. Some problems came for the use of WordNet as well. For example, the sentence "DCR Inc. is looking for Q/A people" won't activate the most general rule in Figure 5. The reason for that is *people* subsumed to concept {*group, grouping*}, but not the concept {*entity*}. This problem can be fixed by adding one more rule with {*group, grouping*} substituting {*entity*} in most general rule in Figure 5. WordNet has very refined senses for each concept, including some rarely used ones, which sometimes causes problems too. This kind of problem certainly hurts the performance, but it's not easy to correct because of the nature of WordNet. However, the use of WordNet generally provides a good method to achieve generalization in this domain of job advertisement.

Conclusion and Future Work

This paper describes a rule optimization approach by using Generalization Tree and WordNet. Our information extraction system learns the necessary knowledge by analyzing sample corpora through a training process. The rule optimization makes it easier for the information extraction system to be customized to a new domain. The Generalization Tree algorithm provides a way to make the system adaptable to the user's needs. The idea of first achieving the highest recall with low precision, then adjusting precision to satisfy user's needs has been successful. We are currently studying how to enhance the system performance by further refining the generalization approach.

References

Aberdeen, John, John Burger, David Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain 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., Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson 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.

Bagga, Amit, and Joyce Y. Chai 1997 A Trainable Message Understanding System *Computational Natural Language Learning (CoNLL97)*,pp. 1-8, July 1997.

Chai, Joyce Y. and Alan W. Biermann 1997 A WordNet Based Rule Generalization Engine For Meaning Extraction *To appear at Tenth International Symposium On Methodologies For Intelligent Systems*, 1997.

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

Church, Kenneth 1988 A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text *Proceedings of the Second Conference on Applied Natural Language Processing*, ACL, 1988.

Church, Kenneth, William Gale, Patrick Hanks, and Donald Hindle. 1989 Parsing, Word Associations and typical Predicate-Argument Relations. *Proceedings of the International Workshop on Parsing Technologies*, 1989.

Derose, S., 1988 Grammatical Category Disambiguation by Statistical Optimization *Computational Linguistics*, 14, 1988.

Fisher, David, Stephen Soderland, Joseph McCarthy, Fangfang Feng and Wendy Lehnert. 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.

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

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

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

Resnik, Philip 1995b Disambiguating noun groupings with respect to WordNet senses, *Third Worshop on Very Large Corpora*, 1995.

Resnik, Philip 1993 Selection and Information: A Class Based Approach to Lexical Relationships, *Ph.D Dissertation, University of Pennsylvania*, 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.