Two Dimensional Generalization in Information Extraction

Joyce Yue Chai
30 Saw Mill River Rd.
IBM T. J. Watson Research Center
Hawthorne, NY 10532
jchai@us.ibm.com

Alan W. Biermann
Computer Science Department
Duke University
Durham, NC 27708
awb@cs.duke.edu

Curry I. Guinn
3400 Cornwallis Rd.
Research Triangle Institute
RTP, NC 27709
cig@rti.org

Abstract

In a user-trained information extraction system, the cost of creating the rules for information extraction can be greatly reduced by maximizing the effectiveness of user inputs. If the user specifies one example of a desired extraction, our system automatically tries a variety of generalizations of this rule including generalizations of the terms and permutations of the ordering of significant words. Where modifications of the rules are successful, those rules are incorporated into the extraction set. The theory of such generalizations and a measure of their usefulness is described.

Introduction

Information extraction (IE) has become a promising area since the advent of the DARPA Message Understanding Conferences (Cowie & Lehnert 1996). Given the vast amount of information available today, successful extraction of useful information has become increasingly important. Most IE systems (MUC6 1995) have used hand-crafted semantic resources for each application domain. However, generation of this domain specific knowledge (i.e., customization) unfortunately requires highly expert computational linguists or developers and is also difficult, time consuming, and prone to error. To address this problem, some researchers have looked at techniques for automatically or semi-automatically constructing lexicons or extraction rules of annotated texts in the domain (Riloff 1993) (Riloff 1996) (Califf & Mooney 1997) (Soderland 1999). Most of these techniques have applied machine learning approaches to learn rules based on texts with filled templates. Either pre-annotation of text is done by a human expert, or the rules are post-processed by a human expert. To keep up with the vast amount of information available for casual users, techniques oriented toward non-experts are desired. By providing an easy training environment to the casual user and by learning rules from the user's input instead of pre-annotated texts, the cost of adapting a system to different domains can be reduced.

In this paper, a new trainable information extraction system is described (Chai 1998). In the new scenario, any user, based on his or her interests, can train the system on different domains. The system learns from the training and automatically outputs the structure target information. A major issue in learning rules concerns the tradeoff between specific, unambiguous extraction rules and the need for general rules that can be applied widely. The balance between those twin goals is essential for good performance.

In building a trainable information extraction system, the cost of creating the rules for information extraction can be greatly reduced by maximizing the effectiveness of user inputs. If the user specifies one example of a desired extraction, our system automatically tries a variety of generalizations of this rule including generalizations of the terms and permutations of the ordering of significant words. Where modifications of the rules are successful, those rules are incorporated into the extraction set. The theory of such generalizations and a measure of their usefulness is given in this paper.

System Overview

Our *T*rainable *Infor**M**ation *E*xtraction *S*ystem (TIMES) includes four major subprocesses: Tokenization, Lexical Processing, Partial Parsing, and Rule Learning and Generalization (Bagga, Chai, & Biermann 1997). The general structure of TIMES is shown in Figure 1. The first stage of processing is carried out by the Tokenizer which segments the input text into sentences and words. Next, the Lexical Process acquires lexical syntactic/semantic information for the words. To achieve this, a Preprocessor is used for semantic classification. The Preprocessor can identify special semantic categories such as email and web addresses, file and directory names, dates, times, and dollar amounts, telephone numbers, zip codes, cities, states, countries, names of companies, and many others. The syntactic information is retrieved from the CELEX database[1] and the semantic information is from WordNet (Miller

[1] CELEX was developed by several universities and institutions in the Netherlands, and is distributed by the Linguistic Data Consortium.

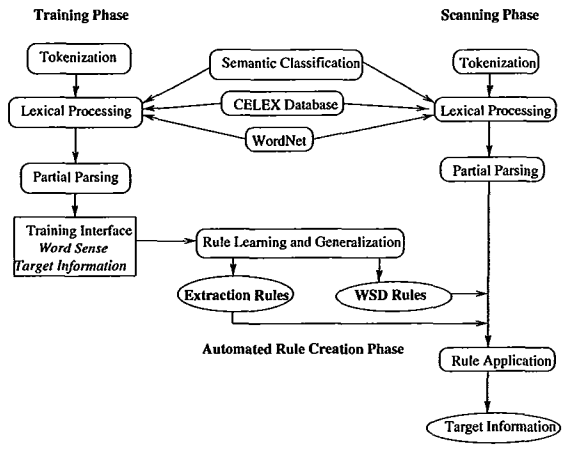

Figure 1: System Overview

1990). Following the Lexical Processing, a Partial Parser, which is based on a set of finite-state rules, is applied to produce a sequence of non-overlapping phrases as output. It identifies noun phrases (NG), verb phrases(VG) and prepositions (PG). The last word of each phrase is identified as its headword. The phrases, together with their syntactic and semantic information are used in Rule Learning and Generalization. Based on the training examples, the system automatically acquires and generalizes a set of rules for future use.

The system has three running modes which implement training, automated rule creation, and scanning, respectively. In the training phase, the user is required to train the system on sample documents based on his/her interests. That is, the user specifically points out the desired information in the text. Because the WordNet hierarchy depends on word senses, in the training phase, the system also requires some semantic tagging by the user. This phase tends to use minimum linguistic and domain knowledge so that it can be used by the casual user. The rule creation phase builds a set of useful rules, including both extraction rules for information of interest and word sense disambiguation (WSD) rules for sense identification. The scanning phase applies the learned rules to any new body of text in the domain.

The Training Phase TIMES provides a convenient training interface for the user. Through this interface, the user identifies the information of interest (i.e., the target information). One issue in training is the trade-off between the user's effort and the system's learning cost. TIMES requires some minimum training from the user. This training identifies the target information for the training articles and assigns the correct sense to the words if their senses are not used as sense one in WordNet. (Sense one is the most frequently used sense in WordNet. The training interface provides sense definitions so that the user would know which sense to choose.) By default, the system will assign sense one to the headwords if no other specific sense is given by the user. If the user performs the minimum training, then the system will learn the rules based on every phrase in the training sentences, and the resulting learning cost will be relatively high. If besides the minimum training, the user decides to select the important phrases from the training examples, then the rules can be learned based only on the important phrases. Thus the training effort is increased, and the system learning cost is reduced. Furthermore, if the user has sufficient expertise and decides to create rules for the system (the rules could be generated from the training articles or from the user's own knowledge), then more training effort is required and the system learning cost is further reduced. In general, to make the system easily and quickly adapted to a new domain, computational linguists and domain experts can apply their expertise if they prefer; casual users can provide the minimum training and rely on the system learning ability.

For example, suppose the training sentence is "The National Technology Group has a need for qualified Inventory Specialists to work at an RTP client site for one month." The user of our system will employ the interface to indicate the key target information in this sentence which is to be extracted. The target information is: companies which have job openings (use *COMPANY* to represent this target), positions available (*POSITION*), locations of those positions (*LOCATION*). Based on the user's input, the system internally generates a record as shown in Table 1. In this table, the first column lists seven important phrases from the sentence; the second column is the target information specified by the user. If a phrase is identified as a type of target information, then this phrase is called *target phrase.* For example, "The National Technology Group," "qualified Inventory Specialist" and "an RTP client site" are three target phrases. The third column lists the specific semantic types classified by the Preprocessor for each important phrase. The fourth column lists the headwords for the important phrases. If a phrase can be identified as a special semantic type, then the headword is the name for that semantic type; otherwise, the headword is the last word in the phrase. The fifth column lists the syntactic categories identified by the Partial Parser. The last column is the sense number for the headwords. From the information in Table 1, the system will create an initial template rule as shown in Figure 2 for doing extraction.

Specific Rules In general, rules in the system are pattern-action rules. The pattern, defined by the left hand side (LHS) of a rule, is a conjunction (expressed by $\wedge$) of *subsumption functions* $S(X, \alpha, target(\alpha))$. X is instantiated by a new phrase when the rule is applied; α is the concept corresponding to the headword of an important phrase in the training sentence; and $target(\alpha)$ is the type of target information identified for α. The action in the right hand side (RHS) of a

important phrases	target	semantic type	headword	syntactic category	sense
The National Technology Group	COMPANY	company_type	company	NG	1
has	none	none	has	VG	1
a need	none	none	need	NG	1
for	none	none	for	PG	1
qualified Inventory Specialists	POSITION	none	specialist	NG	1
at	none	none	at	PG	1
an RTP client site	LOCATION	none	site	NG	1

Table 1: Internal Structure for a Training Example. NG represents noun phrases; VG represents verb phrases and PG represents Prepositions

$S(X_1, \{company\}, COMPANY) \wedge S(X_2, \{has\}, none) \wedge S(X_3, \{need\}, none) \wedge S(X_4, \{for\}, none)$
$\wedge S(X_5, \{specialist\}, POSITION) \wedge S(X_6, \{at\}, none) \wedge S(X_7, \{site\}, LOCATION)$
$\longrightarrow FS(X_1, COMPANY), FS(X_5, POSITION), FS(X_7, LOCATION)$

Figure 2: Initial Template Rule /Most Specific Rule

rule, $FS(X, target(\alpha))$ fills a template slot, or in other words, assigns the type of target information $target(\alpha)$ to the phrase X.

The subsumption function $S(X, \alpha, target(\alpha))$ essentially looks for subsumption of concepts. It returns true if the headword of X is subsumed to the concept α. If all subsumption functions return true, then the RHS actions will take place to extract X as a type $target(\alpha)$. In the following sections, the subsumption function will be referred to as a *rule entity*.

For example, the initial template rule (i.e., the most specific rule) in Figure 2 says that if a pattern of phrases $X_1, X_2, ..., X_7$ is found in a sentence such that the headword of X_1 is subsumed by (or equal to) *company*, the headword of the second phrase X_2 is subsumed by *has*, and so on, then one has found the useful fact that X_1 is a *COMPANY*, X_5 is a *POSITION* and X_7 is a *LOCATION*. (In the most specific rule, the headwords of important phrases are used directly as α in subsumption functions. They are referred to as specific concepts.) Apparently, the initial template rule is very specific and has tremendous limitations. Since the occurrence of the exact same pattern rarely happens in the unseen data, the initial template rule is not very useful. We need to generalize these specific rules.

Rule Generalization

The above sections have shown how the initial template rule is created from a user input. Now we address the issue of how such rules are generalized. In fact, they are generalized by a two dimensional model which is a combination of syntactic (horizontal) and semantic (vertical) generalization.

Syntactic Generalization

From our early experiments, we noticed that the optimum number of entities required in the rules varied for different types of target information. In the job advertisement domain, when a token is classified as the dollar amount semantic type, it is the target salary information 95% of the time. A rule with one rule entity suffices. Rules with more entities are too specific and will lower the performance. For example, in Figure 2, by removing the second, the fourth, the sixth, and the seventh entity, the most specific rule becomes two rules with three entities in Figure 3. Since two target phrases remain and in our convention, each generalized rule only corresponds to one target phrase, two rules are necessary to capture two types of target information. When these two rules are applied for unseen data, the first one will extract the target information *COMPANY* and the second one will extract *POSITION*. On the other hand, since the number of constraints on the LHS are reduced, more target information will be identified. By this observation, removing entities from specific rules results in syntactically generalized rules.

Furthermore, syntactic generalization is also aimed at tackling paraphrase problems. For example, by reordering entities in the rule generated from the training sentence "Fox head Joe Smith ...", it can further process new sentence portions such as "Joe Smith, the head of Fox, ...", "The head of Fox, Joe Smith,...", "Joe Smith, Fox head,..." etc. This type of syntactic generalization is optional in the system. This technique is especially useful when the original specific rules are generated from the user's knowledge.

Therefore, syntactic generalization is designed to learn the appropriate number of entities, and the order of entities in a rule. By reordering and removing rule entities, syntactic generalization is achieved. More precisely, syntactic generalization is attained by a combination function and a permutation function. The combination function selects a subset of rule entities in the most specific rule to form new rules. The permutation function re-orders rule entities to form new rules.

Combination Function If a training sentence has n important phrases, then there are n corresponding

1. $S(X_1, \{company\}, COMPANY) \wedge S(X_2, \{need\}, none) \wedge S(X_3, \{specialist\}, POSITION)$
$\longrightarrow FS(X_1, COMPANY)$
2. $S(X_1, \{company\}, COMPANY) \wedge S(X_2, \{need\}, none) \wedge S(X_3, \{specialist\}, POSITION)$
$\longrightarrow FS(X_3, POSITION)$

Figure 3: An Example of Syntactically Generalized Rules

1. $S(X_1, \{need\}, none) \wedge S(X_2, \{specialist\}, POSITION) \wedge S(X_3, \{company\}, COMPANY)$
$\longrightarrow FS(X_3, COMPANY)$
2. $S(X_1, \{need\}, none) \wedge S(X_2, \{specialist\}, POSITION) \wedge S(X_3, \{company\}, COMPANY)$
$\longrightarrow FS(X_2, POSITION)$

Figure 4: An Example of Permutation Rules

rule entities, $e_1, ..., e_n$ (each e_i is a subsumption function). The combination function selects k rule entities and form the LHS of the rule as $e_{i_1} \wedge e_{i_2} .. \wedge e_{i_k}$ (the order of the entities is the same as the order of the corresponding phrases in the training sentences). At least one of k entities corresponds to a target phrase. If i $(1 < i \leq k)$ entities are created from i target phrases, then i rules will be generated. These rules have the same LHS and different RHS, with each corresponding to one type of target information. Rules created by the combination function are named *combination rules*. For example, in Figure 3, $k = 3$ and $i = 2$, therefore, two rules are necessary to identify two different types of target information.

Permutation Function The permutation function generates new rules by permuting rule entities in the combination rules. This function creates rules for processing paraphrases. Rules created by the permutation function are named *permutation rules*. For example, the permutation function can generate rules in Figure 4 by re-ordering rule entities in Figure 3. While rules in Figure 3 are created based on the sentence "The National Technology Group has a need for qualified Inventory Specialist", the permutation rules can process a paraphrased sentence such as "There is a need for a Inventory Specialist at the National Technology Group."

Semantic Generalization

Syntactic generalization deals with the number and the order of the rule entities. However, each rule entity is still very specific. For a rule entity $S(X, \alpha, target(\alpha))$, semantic generalization is designed to replace α with a more general concept. Thus it will cover more semantically related instances. We use a generic lexical semantic resource, WordNet, for lexical semantic information (Chai & Biermann 1997b).

Concepts in WordNet A concept is defined in WordNet as a set of synonyms (synset). For a word w, the corresponding concept in WordNet is represented by $\{w, w_1, ..., w_n\}$, where each w_i is a synonym of w. Given a word w, its part-of-speech, and sense number, the system can locate a unique corresponding concept in the WordNet hierarchy if w exists in WordNet. For a word, especially a proper noun, if the Preprocessor can identify it as a special semantic type expressed in a concept in WordNet, then a virtual link is created to make the concept of this special noun as the hyponym of that semantic type. For example, "IBM" is not in WordNet, however, it is categorized as a kind of $\{company\}$. The system first creates the concept of "IBM" as $\{IBM\}$, then creates a virtual link between $\{IBM\}$ and $\{company\}$. For any other word w, if it is not in WordNet, and it is not identified as any semantic type, then the concept of w is $\{w\}$ and is virtually put into WordNet, with no hierarchical structure. Therefore, every headword of an important phrase should have a corresponding concept in WordNet.

Let α be a concept in WordNet. The hypernym hierarchical structure provides a path for locating the superordinate concept of α, and it eventually leads to the most general concept above α. (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.)

Therefore, semantic generalization acquires the generalization for each rule entity by replacing the specific concept with a WordNet concept (Chai & Biermann 1997a). For example, rules in Figure 3 could be generalized to rules in Figure 5.

Two Dimensional Generalization

The two-dimensional generalization model is a combination of semantic (vertical) generalization and syntactic (horizontal) generalization. Our method applies a brute force algorithm, which performs semantic generalization on top of syntactically generalized rules generated from the training set, then apply those rules on the training set again. Based on the training examples and a threshold, the system selects useful rules.

The *relevancy_rate* $rel(r_i)$ for rule r_i is defined as the percentage of the correct information extracted by the r_i. A threshold is predefined to control the process. If

1. $S(X_1, \{group, ...\}, COMPANY) \wedge S(X_2, \{need, ...\}, none) \wedge S(X_3, \{professional, ...\}, POSITION)$
$\longrightarrow FS(X_1, COMPANY)$
2. $S(X_1, \{organization, ...\}, COMPANY) \wedge S(X_2, \{need, ...\}, none) \wedge S(X_3, \{engineer, ...\}, POSITION)$
$\longrightarrow FS(X_3, POSITION)$

Figure 5: An Example of Semantically Generalized Rules

$rel(r_i)$ is greater or equal to the threshold, then r_i will be put in the rule base for future use.

The procedure of generating generalized rules is the following:

- Predefine N, the maximum number of entities allowed in the rule.
- For each target phrase in the training sentence, based on the important phrases in the sentence, generate all combinations rules with number of entities from one to N, as well as their permutation rules.
- For every rule, generate all possible semantically generalized rules by replacing each entity with a more general entity with different degree of generalization.
- Apply all rules to the training set. Based on the training examples, compute relevancy_rate for each rule.
- Select rules with relevancy_rate above the defined threshold. If two rules r_1 and r_2 both have n entities, and if each entity of r_1 corresponds to a more general or the same concept as that of r_2, then the system will choose r_1 for future use if $rel(r_1)$ is greater than or equal to the threshold and eliminate r_2.
- Sort the rules with the same number of entities to avoid rule repetition.

By following this procedure, the system will generate a set of useful rules that are both syntactically and semantically generalized. Those rules will be applied to unseen documents. First, the system applies rules with N number of entities to the unseen sentence. If some matches are found, the system identifies the target information as that which is extracted by the most rules and then processes the next sentence. If there is no match, the system applies rules with fewer entities until either there are some matches or all the rules (including those with one entity) have been applied. By doing so, the system will first achieve the highest precision and then gradually increase recall without too much cost in precision. (Precision is the percentage of target information extracted by the system which is correct; recall is the percentage of target information from the text which is correctly extracted by the system.)

Experiments

Our experiments were conducted based on the domain of *triangle.jobs* newsgroup. Eight types of target information were extracted.

- COMPANY (COM.): The name of the company which has job openings.
- POSITION (POS.): The name of the available position.
- SALARY (SAL.): The salary, stipend, compensation information.
- LOCATION (LOC.): The state/city where the job is located.
- EXPERIENCE (EXP.): Years of experience.
- CONTACT (CON.): The phone number or email address for contact.
- SKILLS (SKI.): The specific skills required, such as programming languages, operating systems, etc.
- BENEFITS (BEN.): The benefits provided by the company, such as health, dental insurance,.. etc.

There were 24 articles for training and 40 articles for testing. These 64 articles were randomly selected. Training articles were grouped to three training sets. The first training set contained 8 articles; the second contained 16 articles including ones in the first set; the third training set consisted of all 24 articles. In all of the experiments, the relevancy_rate threshold was set to 0.8 unless specified otherwise. In addition to precision and recall, the F-measure was also computed, which is a combination of precision and recall (Sundheim 1992).

Syntactic Generalization

The first experiment tested the impact of pure syntactic generalization. The system only generated combination rules and permutation rules without semantic generalization. Table 2 shows that rules with pure syntactic generalization (represented by "*all*") achieve better performance (in terms of F-measure) than those rules with one, two or three fixed number of entities.

	one ent.	two ent.	three ent.	all
precision	80.0%	67.4%	67.8%	66.3%
recall	39.8%	54.2%	30.9%	63.9%
F-meas.	53.2%	60.1%	42.5%	65.1%

Table 2: Effect of Rules on All Target Information from Syntactic Generalization

In particular, different types of target information require different numbers of entities in the rules. As shown in Table 3, for the both target information COMPANY and EXPERIENCE, no rule with one entity was learned from the training set. For the target information *SALARY*, the rules with one entity performed much better (in terms of F-measure) than those with

h_{max}	0	1	2	3	4	5	no bound
precision	97.3%	96.5%	96.6%	96.3%	94.0%	92.7%	92.7%
recall	76.8%	77.1%	78.3%	79.2%	80.7%	81.3%	81.3%
F-measure	85.8%	85.7%	86.5%	86.9%	86.8%	86.8%	86.6%

Table 4: Training Set Performance with Respect to Limit of Semantic Generalization

h_{max}	0	1	2	3	4	5	no bound
precision	76.0%	71.0%	59.7%	52.9%	47.1%	44.2%	42.2%
recall	64.6%	67.5%	71.8%	73.3%	74.0%	74.3%	75.8%
F-measure	69.8%	69.2%	65.2%	61.4%	57.6%	55.4%	54.2%

Table 5: Testing Set Performance with Respect to Limit of Semantic Generalization

Fact	meas.	one entity	two entities	three entities	all
COM.	P	0%	76.5%	75.0%	65.9%
	R	0%	68.4%	71.1%	76.3%
	F	0%	72.2%	73.0%	70.7%
POS.	P	94.3%	71.9%	64.3%	76.4%
	R	51.6%	35.9%	14.1%	60.9%
	F	66.7%	47.9%	23.1%	67.8%
SAL.	P	91.7%	84.6%	100%	88.9%
	R	68.8%	34.4%	12.5%	75.0%
	F	78.6%	48.9%	28.1%	81.4%
LOC.	P	16.7%	33.7%	37.2%	32.0%
	R	2.7%	41.1%	21.9%	42.5%
	F	4.6%	37.0%	27.6%	36.5%
EXP.	P	0%	54.8%	53.6%	48.6%
	R	0%	54.8%	48.4%	54.8%
	F	0%	54.8%	50.9%	51.5%
CON.	P	97.6%	87.3%	91.7%	88.4%
	R	51.9%	71.4%	14.3%	79.2%
	F	67.8%	78.6%	24.7%	83.7%
SKI.	P	74.1%	75.0%	76.4%	71.7%
	R	61.3%	64.4%	41.7%	70.0%
	F	67.1%	69.3%	54.0%	70.8%
BEN.	P	90.0%	93.3%	100%	94.1%
	R	22.5%	35.0%	25.0%	40.0%
	F	36.0%	50.9%	40.0%	56.1%

Table 3: Effect of Rules on Target Information from Syntactic Generalization, where P is precision, R is recall, and F is F-measure.

two entities or three entities. For the target information *LOCATION*, the rules with two entities performed better than those of one entity and three entities. For the target information *COMPANY*, the rules with three entities perform the best. Thus, for different types of information, the best extraction requires different numbers of entities in the rules. However, the appropriate number of entities for each type of information is not known in advance. If applying rules with the fixed number of entities, a less than optimal number of entities will cause a significant loss in the performance. By applying our approach, for each type of information, the performance is either better (as in the SALARY row), or slightly worse (as in the COMPANY row). The overall performance of the application algorithm is better than that achieved by the rules with the fixed number of entities.

Two Dimensional Generalization

When the semantic generalization is added to the model, the evaluation of the effectiveness of WordNet becomes important. In this experiment, the system generated rules both syntactically and semantically. In those rules, some entities were only generalized to include the synonyms of the specific concept from the training sentence; some were generalized to direct hypernyms (one level above the specific concept in the conceptual hierarchy), and some were generalized to various higher degrees. The question is, even though the rules have been learned from the training examples, are they reliable? Do we need to put some upper bound on the generalization degree for the entities in order to achieve good performance? To answer that, we used h_{max} as the limit for the degree of generalization for each entity. We modified the rules to only generalize each entity to h_{max} level above the specific concept in WordNet hierarchy. We applied those rules (the threshold was 1.0) with different limitations on semantic generalization to the 24 training documents and 40 testing documents.

As in Table 4, for the training set, with no upper bound on semantic generalization degree in the rules, the system had an overall 86.6% F-measure, with very high (92.7%) precision. When the various limits on the generalization degree were applied, the performance was still about the same. This indicated that the rules were indeed learned from the training examples, and these rules sufficiently represented the training examples.

We then applied the same set of rules on the testing data. As shown in Table 5, without an upper bound on the semantic generalization, the system only achieved an F-measure of 54.2%. However, when the upper bound on the degree of generalization was $h_{max} = 0$,

which only generalized the entities to include the synonyms of the specific concepts, the overall performance was about 70%. The results indicated further restriction on the semantic generalization degree could enhance the performance. WordNet hypernyms are useful in achieving high recall, but with a high cost in precision. WordNet synonyms and direct hypernyms are particularly useful in balancing the tradeoff between precision and recall and thus improving the overall performance.

training articles	8	16	24
precision	61.5%	66.7%	66.3%
recall	48.8%	61.8%	63.9%
F-measure	54.4%	64.2%	65.1%

Table 6: Performance vs. Training Effort from Syntactic Generalization

training articles	8	16	24
precision	68.8%	70.2%	71.0%
recall	59.8%	65.2%	67.5%
F-measure	64.0%	68.0%	69.2%

Table 7: Performance vs. Training Effort from Two Dimensional Generalization

We compared the experimental results from rules with pure syntactic generalization and the rules with two dimensional generalization (with a limit of $h_{max} = 1$ on the semantic generalization). As shown in Table 6 and Table 7, when the training set is small, the semantic generalization can be particularly helpful. The F-measure increased about 10% when the training set only had eight articles. The F-measure for training 16 articles is about the same as that from training 24 articles. Thus no more training is necessary. This result implies that for the two-dimensional rule generalization approach, there is a performance upper bound in this domain. If we would like to break this upper bound, generalizing only concepts and orders of the rules may not be enough. We should approach other strategies for generalization. For example, verb forms could be generalized to verb nominalization forms.

The semantic generalization can be particularly effective for extracting certain types of information. Comparing Table 3 and Table 8, we can see that the semantic generalization is especially useful for extracting both LOCATION and BENEFITS facts. The performance was improved about 30% in F-measure for those two facts.

Discussion

Automated rule learning from examples can also be found in other systems such as AutoSlog (Riloff 1993), PALKA (Kim & Moldovan 1993), RAPIER (Califf & Mooney 1997) and WHISK (Soderland 1999). AutoSlog

Fact	meas.	one entity	two entity	three entity	all
COM.	P	0%	71.8%	71.4%	67.4%
	R	0%	75.7%	81.1%	83.8%
	F	0%	73.7%	75.9%	74.7%
POS.	P	94.3%	69.8%	57.1%	72.3%
	R	51.6%	46.9%	25.0%	53.1%
	F	66.7%	56.1%	34.8%	61.2%
SAL.	P	92.0%	84.6%	100%	88.5%
	R	71.9%	34.4%	12.5%	71.9%
	F	80.7%	48.9%	22.2%	80.2%
LOC.	P	25.0%	63.3%	66.7%	59.4%
	R	1.4%	78.1%	63.0%	78.1%
	F	2.7%	69.9%	64.8%	67.5%
EXP.	P	0%	42.9%	51.4%	40.4%
	R	0%	54.5%	57.6%	57.6%
	F	0%	48.0%	54.3%	47.5%
CON.	P	83.3%	86.4%	78.6%	81.8%
	R	51.9%	74.0%	14.3%	81.8%
	F	64.0%	79.7%	27.1%	81.8%
SKI.	P	33.3%	66.9%	67.9%	63.0%
	R	1.9%	71.4%	57.8%	70.8%
	F	3.59%	69.1%	62.4%	66.7%
BEN.	P	95.8%	95.7%	100%	96.8%
	R	57.5%	55.0%	42.5%	75.0%
	F	71.9%	69.9%	60.0%	84.5%

Table 8: Effect of Rules on Target Information from Two Dimensional Generalization

uses heuristics to create rules from the examples and then requires the human expert to accept or reject the rules. PALKA applies a conceptual hierarchy to control the generalization or specification of the target slot. RAPIER uses inductive logic programming to learn the patterns that characterizes slot-fillers and their context. WHISK learns the rules by starting with a seed example and then selectively adding terms that appear in the seed example to a rule.

TIMES differentiates itself in the sense that it learns rules by interleaving syntactic generalization and semantic generalization. It automatically decides the number, the order and the generalization/specification of constraints. It learns these three aspects in each rule while other systems concentrate on one or two aspects. Furthermore, most of those systems focus on the improvement of performance by refining learning techniques in an environment of large databases of examples. TIMES is designed to provide a paradigm where rule learning makes it possible to build an IE system based on minimum training by a casual user. Indeed, TIMES emphasizes the usability to the casual user. When a large amount of pre-annotated information is not available and when the user is not experienced enough to tag the information, how does one make an IE system effective based on minimum training? In our experiments, we intentionally chose a small training set since for a casual user, large amount of

training is difficult. The experimental results suggest that the two dimensional generalization obtains reasonable performance while the effort and time involved in the training is dramatically reduced.

Most information extraction systems are created based on hand-crafted domain specific knowledge. This is because the lexical semantic definitions given by the generic resources sometimes cannot meet the actual needs of the specific domain. No use is made of existing general lexical semantic resources by any of the MUC systems. NYU's MUC-4 system (Grishman, Macleod, & Sterling 1992) made some attempt at using WordNet for semantic classification. However, they ran into the problem of automated sense disambiguation because the WordNet hierarchy is sense dependent. As a result, they gave up using WordNet. TIMES attempts to integrate WordNet with WSD techniques (Chai & Biermann 1999). The use of WordNet hypernyms in the two-dimensional generalization could raise the recall performance from 65% to 76% (see Table 5) at the cost of precision. However, we found that the WordNet synonyms and the direct hypernyms are particularly useful in balancing the tradeoff between the precision and recall. The use of WordNet enhanced overall performance by 5%. Despite the typographical errors, incorrect grammars and rare abbreviations in the free text collection which make information extraction more difficult, in this domain, the two-dimensional generalization model based on both syntactic generalization and semantic generalization achieved about 69% (see Table 7) F-measure in overall performance.

Conclusion

This paper presents the basic machinery for creating an efficient synthesizer of information extraction systems. The user can start with a basic system and an extraction problem in mind and create the rules necessary to do the extraction from hand-entered examples. If the problem is no more difficult than the domain described here, then only eight to sixteen example articles must be hand-processed to obtain F-measure in the 60-70 percent range. With a well designed GUI, these examples can be done in just a few minutes each, and the total user investment will be approximately two hours. The result will be a set of rules that have numbers of entities, orderings of the entities, and levels of generalization automatically specialized to optimally extract each fact of interest from a large database.

Acknowledgements

We would like to thank Amit Bagga for developing Tokenizer and Preprocessor. We would also like to thank Jerry Hobbs for providing us with the finite state rules for the Partial Parser. This work was supported in part by an IBM Fellowship.

References

Bagga, A.; Chai, J.; and Biermann, A. 1997. The role of WordNet in the creation of a trainable message understanding system. *Proceedings of Ninth Conference on Innovative Applications of Artificial Intelligence (IAAI-97).*

Califf, M., and Mooney, R. 1997. Relational learning of pattern-match rules for information extraction. *Proceedings of Computational Language Learning'97.*

Chai, J., and Biermann, A. 1997a. Corpus based statistical generalization tree in rule optimization. *Proceedings of Fifth Workshop on Very Large Corpora (WVLC-5).*

Chai, J., and Biermann, A. 1997b. A WordNet based rule generalization engine for meaning extraction. *Lecture Notes in Artificial Intelligence (1325): Foundations of Intelligent Systems.*

Chai, J., and Biermann, A. 1999. The use of word sense disambiguation in an information extraction system. *Proceedings of Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99).*

Chai, J. 1998. *Learning and Generalization in the Creation of Information Extraction Systems.* Ph.D. Dissertation, Department of Computer Science, Duke University.

Cowie, J., and Lehnert, W. 1996. Information extraction. *Communications of ACM.*

Grishman, R.; Macleod, C.; and Sterling, J. 1992. New York University Proteus System: MUC-4 test results and analysis. *Proceedings of the Fourth Message Understanding Conference.*

Kim, J., and Moldovan, D. 1993. Acquisition of semantic patterns for information extraction from corpora. *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications.*

Miller, G. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography.*

MUC6. 1995. *Proceedings of the Sixth Message Understanding Conference.*

Riloff, E. 1993. Automatically constructing a dictionary for information extraction tasks. *Proceedings of the Eleventh National Conference on Artificial Intelligence.*

Riloff, E. 1996. An empirical study of automated dictionary construction for information extraction in three domains. *AI Journal.*

Soderland, S. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning Journal Special Issues on Natural Language Learning.*

Sundheim, B. 1992. Overview of the fourth message understanding evaluation and conference. *Proceedings of Fourth Message Understanding Conference (MUC-4).*