Closing the Gap: Learning-Based Information Extraction Rivaling Knowledge-Engineering Methods

Hai Leong Chieu
DSO National Laboratories
20 Science Park Drive
Singapore 118230
chaileon@dso.org.sg

Hwee Tou Ng
Department of Computer Science
National University of Singapore
3 Science Drive 2
Singapore 117543
nght@comp.nus.edu.sg

Yoong Keok Lee
DSO National Laboratories
20 Science Park Drive
Singapore 118230
lyoongke@dso.org.sg

Abstract

In this paper, we present a learning approach to the scenario template task of information extraction, where information filling one template could come from multiple sentences. When tested on the MUC-4 task, our learning approach achieves accuracy competitive to the best of the MUC-4 systems, which were all built with manually engineered rules. Our analysis reveals that our use of full parsing and state-of-the-art learning algorithms have contributed to the good performance. To our knowledge, this is the first research to have demonstrated that a learning approach to the full-scale information extraction task could achieve performance rivaling that of the knowledge-engineering approach.

1 Introduction

The explosive growth of online texts written in natural language has prompted much research into *information extraction* (IE), the task of automatically extracting specific information items of interest from natural language texts. The extracted information is used to fill database records, also known as *templates* in the IE literature.

Research efforts on IE tackle a variety of tasks. They include extracting information from semi-structured texts, such as seminar announcements, rental and job advertisements, etc., as well as from free texts, such as newspaper articles (Soderland, 1999). IE from semi-structured texts is easier than from free texts, since the layout and format of a semi-structured text provide additional useful clues to aid in extraction. Several benchmark data sets have been used to evaluate IE approaches on semi-structured texts (Soderland, 1999; Ciravegna, 2001; Chieu and Ng, 2002a).

AYACUCHO, 19 JAN 89 – TODAY TWO PEOPLE WERE WOUNDED WHEN A BOMB EXPLODED IN SAN JUAN BAUTISTA MUNICIPALITY. OFFICIALS SAID THAT SHINING PATH MEMBERS WERE RESPONSIBLE FOR THE ATTACK ... ... POLICE SOURCES STATED THAT THE BOMB ATTACK INVOLVING THE SHINING PATH CAUSED SERIOUS DAMAGES ... ...

Figure 1: Snippet of a MUC-4 document

For the task of extracting information from free texts, a series of Message Understanding Conferences (MUC) provided benchmark data sets for evaluation. Several subtasks for IE from free texts have been identified. The named entity (NE) task extracts person names, organization names, location names, etc. The template element (TE) task extracts information centered around an entity, like the acronym, category, and location of a company. The template relation (TR) task extracts relations between entities. Finally, the full-scale IE task, the scenario template (ST) task, deals with extracting generic information items from free texts. To tackle the full ST task, an IE system needs to merge information from multiple sentences in general, since the information needed to fill one template can come from multiple sentences, and thus discourse processing is needed. The full-scale ST task is considerably harder than all the other IE tasks or subtasks outlined above.

As is the case with many other natural language processing (NLP) tasks, there are two main approaches to IE, namely the knowledge-engineering approach and the learning approach. Most early IE systems adopted the knowledge-engineering ap-

0	MESSAGE: ID	TST3-MUC4-0014
1	MESSAGE: TEMPLATE	1
2	INCIDENT: DATE	19-JAN-89
3	INCIDENT: LOCATION	PERU: SAN JUAN BAUTISTA (MUNICIPALITY)
4	INCIDENT: TYPE	BOMBING
5	INCIDENT: STAGE OF EXECUTION	ACCOMPLISHED
6	INCIDENT: INSTRUMENT ID	"BOMB"
7	INCIDENT: INSTRUMENT TYPE	BOMB:"BOMB"
8	PERP: INCIDENT CATEGORY	TERRORIST ACT
9	PERP: INDIVIDUAL ID	"SHINING PATH MEMBERS"
10	PERP: ORGANIZATION ID	"SHINING PATH"
11	PERP: ORGANIZATION CONFIDENCE	SUSPECTED OR ACCUSED BY AUTHORITIES:"SHINING PATH"
12	PHYS TGT: ID	-
13	PHYS TGT: TYPE	-
14	PHYS TGT: NUMBER	-
15	PHYS TGT: FOREIGN NATION	-
16	PHYS TGT: EFFECT OF INCIDENT	SOME DAMAGE:"-"
17	PHYS TGT: TOTAL NUMBER	-
18	HUM TGT: NAME	-
19	HUM TGT: DESCRIPTION	"PEOPLE"
20	HUM TGT: TYPE	CIVILIAN:"PEOPLE"
21	HUM TGT: NUMBER	2:"PEOPLE"
22	HUM TGT: FOREIGN NATION	-
23	HUM TGT: EFFECT OF INCIDENT	INJURY:"PEOPLE"
24	HUM TGT: TOTAL NUMBER	-

Figure 2: Example of a MUC-4 template

proach, where manually engineered rules were used for IE. More recently, machine learning approaches have been used for IE from semi-structured texts (Califf and Mooney, 1999; Soderland, 1999; Roth and Yih, 2001; Ciravegna, 2001; Chieu and Ng, 2002a), named entity extraction (Chieu and Ng, 2002b), template element extraction, and template relation extraction (Miller et al., 1998). These machine learning approaches have been successful for these tasks, achieving accuracy comparable to the knowledge-engineering approach.

However, for the full-scale ST task of generic IE from free texts, the best reported method to date is still the knowledge-engineering approach. For example, almost all participating IE systems in MUC used the knowledge-engineering approach for the full-scale ST task. The one notable exception is the work of UMass at MUC-6 (Fisher et al., 1995). Unfortunately, their learning approach did considerably worse than the best MUC-6 systems. Soderland (1999) and Chieu and Ng (2002a) attempted machine learning approaches for a scaled-down version of the ST task, where it was assumed that the information needed to fill one template came from one sentence only.

In this paper, we present a learning approach to the full-scale ST task of extracting information from free texts. The task we tackle is considerably more complex than that of (Soderland, 1999; Chieu and Ng, 2002a), since we need to deal with merging information from multiple sentences to fill one template. We evaluated our learning approach on the MUC-4 task of extracting terrorist events from free texts. We chose the MUC-4 task since manually prepared templates required for training are available.[1] When trained and tested on the official benchmark data of MUC-4, our learning approach achieves accuracy competitive with the best MUC-4 systems, which were all built using manually engineered rules. To our knowledge, our work is the first learning-based approach to have achieved performance competitive with the knowledge-engineering approach on the full-scale ST task.

2 Task Definition

The task addressed in this paper is the Scenario Template (ST) task defined in the Fourth Message Understanding Conference (MUC-4).[2] The objective of this task is to extract information on terrorist events occurring in Latin American countries from free text documents. For example, given the input document in Figure 1, an IE system is to extract information items related to any terrorist events to fill zero or more database records, or *templates*. Each distinct terrorist event is to fill one template. An example of an output template is shown in Figure 2. Each of the 25 fields in the template is called a *slot*, and the string or value that fills a slot is called a *slot fill*.

Different slots in the MUC-4 template need to be treated differently. Besides slot 0 (MESSAGE: ID) and slot 1 (MESSAGE: TEMPLATE), the other 23 slots have to be extracted or inferred from the text document. These slots can be divided into the following categories:

String Slots. These slots are filled using strings extracted directly from the text document (slot 6, 9, 10, 12, 18, 19).

Text Conversion Slots. These slots have to be inferred from strings in the document (slot 2, 14, 17, 21, 24). For example, INCIDENT: DATE has to be inferred from temporal expressions such as "TO-

[1] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/muc_data/muc_data_index.html

[2] The full-scale IE task is called the ST task only in MUC-6 and MUC-7, when other subtasks like NE and TE tasks were defined. Here, we adopted this terminology also in describing the full-scale IE task for MUC-4.

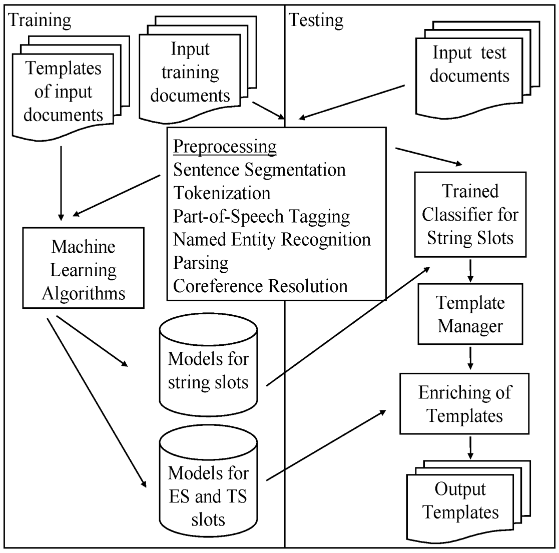

Figure 3: ALICE: our information extraction system

DAY", "LAST WEEK", etc.

Set Fill Slots. This category includes the rest of the slots. The value of a set fill slot comes from a finite set of possible values. They often have to be inferred from the document.

3 The Learning Approach

Our supervised learning approach is illustrated in Figure 3. Our system, called ALICE (Automated Learning-based Information Content Extraction), requires manually extracted templates paired with their corresponding documents that contain terrorist events for training. After the training phase, ALICE is then able to extract relevant templates from new documents, using the model learnt during training.

In the training phase, each input training document is first preprocessed through a chain of preprocessing modules. The outcome of the preprocessing is a full parse tree for each sentence, and coreference chains linking various coreferring noun phrases both within and across sentences. The core of ALICE uses supervised learning to build one classifier for each string slot. The candidates to fill a template slot are base (non-recursive) noun phrases. A noun phrase np that occurs in a training document d and fills a template slot s is used to generate one positive training example for the classifier of slot s. Other noun phrases in the training document d are negative training examples for the classifier of slot s. The features of a training example generated from np are the verbs and other noun phrases (serving roles like agent and patient) related to np in the same sentence, as well as similar features for coreferring noun phrases of np. Thus, our features for a template slot classifier encode semantic (agent and patient roles) and discourse (coreference) information. Our experimental results in this paper demonstrate that such features are effective in learning what to fill a template slot.

During testing, a new document is preprocessed through the same chain of preprocessing modules. Each candidate noun phrase np generates one test example, and it is presented to the classifier of a template slot s to determine whether np fills the slot s. A separate template manager decides whether a new template should be created to include slot s, or slot s should fill the existing template.

3.1 Preprocessing

All the preprocessing modules of ALICE were built with supervised learning techniques. They include sentence segmentation (Ratnaparkhi, 1998), part-of-speech tagging (Charniak et al., 1993), named entity recognition (Chieu and Ng, 2002b), full parsing (Collins, 1999), and coreference resolution (Soon et al., 2001). Each module performs at or near state-of-the-art accuracy, but errors are unavoidable, and later modules in the preprocessing chain have to deal with errors made by the previous modules.

3.2 Features in Training and Test Examples

As mentioned earlier, the features of an example are generated based on a base noun phrase (denoted as baseNP), which is a candidate for filling a template slot. While most strings that fill a string slot are base noun phrases, this is not always the case. For instance, consider the two examples in Figure 4. In the first example, "BOMB" should fill the string slot INCIDENT: INSTRUMENT ID, while in the second example, "FMLN" should fill the string slot PERP: ORGANIZATION ID. However, "BOMB" is itself not a baseNP (the baseNP is "A BOMB EXPLOSION"). Similarly for "FMLN".

As such, a string that fills a template slot but is itself not a baseNP (like "BOMB") is also used to generate a training example, by using its smallest encompassing noun phrase (like "A BOMB EXPLO-

(1) ONE PERSON WAS KILLED TONIGHT AS THE RESULT OF A BOMB EXPLOSION IN SAN SALVADOR.
(2) FORTUNATELY, NO CASUALTIES WERE REPORTED AS A RESULT OF THIS INCIDENT, FOR WHICH THE FMLN GUERRILLAS ARE BEING HELD RESPONSIBLE.

Figure 4: Sentences illustrating string slots that cannot be filled by baseNPs.

(1) MEMBERS OF THAT SECURITY GROUP ARE COMBING THE AREA TO DETERMINE THE FINAL OUTCOME OF THE FIGHTING.
(2) A BOMB WAS THROWN AT THE HOUSE OF FREDEMO CANDIDATE FOR DEPUTY MIGUEL ANGEL BARTRA BY TERRORISTS.

Figure 5: Sample sentences for the illustration of features

SION”) to generate the training example features. During training, a list of such words is compiled for slots 6 and 10 from the training templates. During testing, these words are also used as candidates for generating test examples for slots 6 and 10, in addition to base NPs.

The features of an example are derived from the treebank-style parse tree output by an implementation of Collins' parser (Collins, 1999). In particular, we traverse the full parse tree to determine the verbs, agents, patients, and indirect objects related to a noun phrase candidate np. While a machine learning approach is used in (Gildea and Jurafsky, 2000) to determine general semantic roles, we used a simple rule-based traversal of the parse tree instead, which could also reliably determine the generic agent and patient role of a sentence, and this suffices for our current purpose.

Specifically, for a given noun phrase candidate np, the following groups of features are used:

Verb of Agent NP (VAg) When np is an agent in a sentence, each of its associated verbs is a VAg feature. For example, in sentence (1) of Figure 5, if np is MEMBERS, then its VAg features are COMB and DETERMINE.

Verb of Patient NP (VPa) When np is a patient in a sentence, each of its associated verbs is a VPa feature. For example, in sentence (2) of Figure 5, if np is BOMB, then its VPa feature is THROW.

Verb-Preposition of NP-in-PP (V-Prep) When np is the NP in a prepositional phrase PP, then this feature is the main verb and the preposition of PP. For example, in sentence (2) of Figure 5, if np is HOUSE, its V-Prep feature is THROW-AT.

VPa and related NPs/PPs (VPaRel) If np is a patient in a sentence, each of its VPa may have its own agents (Ag) and prepositional phrases (Prep-NP). In this case, the tuples (VPa, Ag) and (VPa, Prep-NP) are used as features. For example, in “GUARDS WERE SHOT TO DEATH”, if np is GUARDS, then its VPa SHOOT, and the prepositional phrase TO-DEATH form the feature (SHOOT, TO-DEATH).

VAg and related NPs/PPs (VAgRel) This is similar to VPa above, but for VAg.

V-Prep and related NPs (V-PrepRel) When np is the NP in a prepositional phrase PP, then the main verb (V) may have its own agents (Ag) and patients (Pa). In this case, the tuples (Ag, V-Prep) and (V-Prep, Pa) are used as features. For example, HOUSE in sentence (2) of Figure 5 will have the features (TERRORIST, THROW-AT) and (THROW-AT, BOMB).

Noun-Preposition (N-Prep) This feature aims at capturing information in phrases such as “MURDER OF THE PRIESTS”. If np is PRIESTS, this feature will be MURDER-OF.

Head Word (H) The head word of each np is also used as a feature. In a parse tree, there is a head word at each tree node. In cases where a phrase does not fit into a parse tree node, the last word of the phrase is used as the head word. This feature is useful as the system has no information of the semantic class of np. From the head word, the system can get some clue to help decide if np is a possible candidate for a slot. For example, an np with head word PEASANT is more likely to fill the human target slot compared to another np with head word CLASH.

Named Entity Class (NE) The named entity class of np is used as a feature.

Real Head (RH) For a phrase that does not fit into a parse node, the head word feature is taken to be the last word of the phrase. The real head word of its encompassing parse node is used as another feature. For example, in the NP “FMLN GUERRILLAS”, “FMLN” is a positive example for slot 10, with head word “FMLN” and real head “GUERRILLA”.

Coreference features Coreference chains found by our coreference resolution module based on decision tree learning are used to determine the noun phrases that corefer with np. In particular, we use the two noun phrases np_{-1} and np_{+1}, where np_{-1} (np_{+1}) is the noun phrase that corefers with np and immediately precedes (follows) np. If such a preceding (or following) noun phrase np' exists, we generate the following features based on np': VAg, VPa, and N-Prep.

To give an idea of the informative features used in the classifier of a slot, we rank the features used for a slot classifier according to their correlation metric values (Chieu and Ng, 2002a), where informative features are ranked higher. Table 1 shows the top-ranking features for a few feature groups and template slots. The bracketed number behind each feature indicates the rank of this feature for that slot classifier, ordered by the correlation metric value. We observed that certain feature groups are more useful for certain slots. For example, DIE is the top VAg verb for the human target slot, and is ranked 12 among all features used for the human target slot. On the other hand, VAg is so unimportant for the physical target slot that the top VAg verb is due to a preprocessing error that made MONSERRAT a verb.

3.3 Supervised Learning Algorithms

We evaluated four supervised learning algorithms.

Maximum Entropy Classifier (Alice-ME)

The maximum entropy (ME) framework is a recent learning approach which has been successfully used in various NLP tasks such as sentence segmentation, part-of-speech tagging, and parsing (Ratnaparkhi, 1998). However, to our knowledge, ours is the first research effort to have applied ME learning to the full-scale ST task. We used the implementation of maximum entropy modeling from the opennlp.maxent package.[3].

Support Vector Machine (Alice-SVM) The Support Vector Machine (SVM) (Vapnik, 1995) has been successfully used in many recent applications such as text categorization and handwritten digit recognition. The learning algorithm finds a hyperplane that separates the training data with the largest margin. We used a linear kernel for all our experiments.

Naive Bayes (Alice-NB) The Naive Bayes (NB) algorithm (Duda and Hart, 1973) assumes the independence of features given the class and assigns a test example to the class which has the highest posterior probability. Add-one smoothing was used.

Decision Tree (Alice-DT) The decision tree (DT) algorithm (Quinlan, 1993) partitions training examples using the feature with the highest information gain. It repeats this process recursively for each partition until all examples in each partition belong to one class.

We used the WEKA package[4] for the implementation of SVM, NB, and DT algorithms.

A feature cutoff c is used for each algorithm: features occurring less than c times are rejected. For all experiments, c is set to 3. For ME and SVM, no other feature selection is applied. For NB and DT, the top 100 features as determined by chi-square are selected. While not trying to do a serious comparison of machine learning algorithms, ME and SVM seem to be able to perform well without feature selection, whereas NB and DT require some form of feature selection in order to perform reasonably well.

3.4 Template Manager

As each sentence is processed, phrases classified as positive for any of the string slots are sent to the Template Manager (TM), which will decide if a new template should be created when it receives a new slot fill.

The system first attempts to attach a date and a location to each slot fill np. Dates and locations are first attached to their syntactically nearest verb, by traversing the parse tree. Then, for each string fill np, we search its syntactically nearest verb v in the same manner and assign the date and location attached to v to np.

When a new slot fill is found, the Template Manager will decide to start a new template if one of the following conditions is true:

Date The date attached to the current slot fill is different from the date of the current template.

Location The location attached to the current slot fill is not compatible with the location of the current template (one location does not contain the other).

[3] http://maxent.sourceforge.net

[4] http://www.cs.waikato.ac.nz/ml/weka

Slot	VAg	VPa	V-Prep	N-Prep
Human Target	DIE(12)	KILL(2)	IDENTIFY-AS(47)	MURDER-OF(3)
Perpetrator Individual	KIDNAP(5)	IMPLICATE(17)	ISSUE-FOR(73)	WARRANT-FOR(64)
Physical Target	MONSERRAT(420)	DESTROY(1)	THROW-AT(32)	ATTACK-ON(11)
Perpetrator Organization	KIDNAP(16)	BLAME(25)	SUSPEND-WITH(87)	GUERRILLA-OF(31)
Instrument ID	EXPLODE(4)	PLACE(5)	EQUIP-WITH(31)	EXPLOSION-OF(17)

Table 1: The top-ranking feature for each group of features and the classifier of a slot

Incident Type	Seed Words
ATTACK	JESUIT, MURDER, KILL, ATTACK
BOMBING	BOMB, EXPLOS, DYNAMIT, EXPLOD, INJUR
KIDNAPPING	KIDNAP, ELN, RELEAS

Table 2: Stemmed seed words for each incident type

This is determined by using location lists provided by the MUC-4 conference, which specify whether one location is contained in another. An entry in this list has the format of “PLACE-NAME1:PLACE-NAME2”, where PLACE-NAME2 is contained in PLACE-NAME1 (e.g., CUBA: HAVANA (CITY)).

Seed Word The sentence of the current slot fill contains a seed word for a different incident type. A number of seed words are automatically learned for each of the incident types ATTACK, BOMBING, and KIDNAPPING. They are automatically derived based on the correlation metric value used in (Chieu and Ng, 2002a). For the remaining incident types, there are too few incidents in the training data for seed words to be collected. The seeds words used are shown in Table 2.

3.5 Enriching Templates

In the last stage before output, the template content is further enriched in the following manner:

Removal of redundant slot fills For each slot in the template, there might be several slot fills referring to the same thing. For example, for HUM TGT: DESCRIPTION, the system might have found both “PRIESTS” and “JESUIT PRIESTS”. A slot fill that is a substring of another slot fill will be removed from the template.

Effect/Confidence and Type Classifiers are also trained for effect and confidence slots 11, 16, and 23 (ES slots), as well as type slots 7, 13, and 20 (TS slots). ES slots used exactly the same features as string slots, while TS slots used only head words and adjectives as features. For such slots, each entry refers to another slot fill. For example, slot 23 may contain the entry “DEATH” : “PRIESTS”, where “PRIESTS” fills slot 19. During training, each training example is a fill of a reference slot (e.g., for slot 23, the reference slots are slot 18 and 19). For slot 23, for example, each instance will have a class such as DEATH or INJURY, or if there is no entry in slot 23, UNKNOWN_EFFECT. During testing, slot fills of reference slots will be classified to determine if they should have an entry in an ES or a TS slot.

Date and Location. If the system is unable to fill the DATE or LOCATION slot of a template, it will use as default value the date and country of the city in the dateline of the document.

Other Slots. The remaining slots are filled with default values. For example, slot 5 has the default value “ACCOMPLISHED”, and slot 8 “TERRORIST ACT” (except when the perpetrator contains strings such as “GOVERNMENT”, in which case it will be changed to “STATE-SPONSORED VIOLENCE”). Slot 15, 17, 22, and 24 are always left unfilled.

4 Evaluation

There are 1,300 training documents, of which 700 are relevant (i.e., have one or more event templates). There are two official test sets, i.e., TST3 and TST4, containing 100 documents each. We trained our system ALICE using the 700 documents with relevant templates, and then tested it on the two official test sets. The output templates were scored using the scorer provided on the official website.

The accuracy figures of ALICE (with different learning algorithms) on string slots and all slots are listed in Table 3 and Table 4, respectively. Accuracy is measured in terms of recall (R), precision (P), and F-measure (F). We also list in the two tables the accuracy figures of the top 7 (out of a total of 17) systems that participated in MUC-4. The accuracy figures in the two tables are obtained by running the official scorer on the output templates of ALICE, and those of the MUC-4 participating systems (available

	TST3				TST4		
	R	P	F		R	P	F
GE	55	54	54	GE	60	54	57
GE-CMU	43	52	47	GE-CMU	48	52	50
Alice-ME	41	51	45	Alice-ME	44	49	46
Alice-SVM	41	45	43	Alice-SVM	45	44	44
SRI	37	51	43	NYU	42	45	43
UMASS	36	49	42	SRI	39	49	43
Alice-DT	31	51	39	Alice-DT	36	50	42
NYU	35	43	39	UMASS	42	42	42
Alice-NB	41	30	35	Alice-NB	51	32	39
UMICH	32	36	34	BBN	35	42	38
BBN	22	40	28	UMICH	32	34	33

Table 3: Accuracy of string slots on the TST3 and TST4 test set

	TST3				TST4		
	R	P	F		R	P	F
GE	58	54	56	GE	62	53	57
GE-CMU	48	55	51	GE-CMU	53	53	53
UMASS	45	56	50	SRI	44	51	47
Alice-ME	46	51	48	Alice-ME	46	46	46
SRI	43	54	48	NYU	46	46	46
Alice-SVM	45	46	45	UMASS	47	45	46
Alice-DT	38	53	44	Alice-SVM	47	40	43
NYU	40	46	43	Alice-DT	41	46	43
UMICH	40	39	39	BBN	40	43	41
Alice-NB	45	34	39	Alice-NB	52	33	40
BBN	29	43	35	UMICH	36	34	35

Table 4: Accuracy of all slots on the TST3 and TST4 test set

Test set/slots	Systems in the same group
TST3/string	GE-CMU, SRI, UMASS, NYU
TST4/string	GE-CMU, Alice-SVM, NYU, SRI, Alice-DT, UMASS
TST3/all	GE-CMU, UMASS, SRI, NYU
TST4/all	SRI, NYU, UMASS, Alice-SVM, Alice-DT, BBN

Table 5: Systems whose F-measures are not significantly different from Alice-ME at the 0.10 significance level with 0.99 confidence

	TST3			TST4		
System	R	P	F	R	P	F
H + NE	23	44	30	18	30	23
H + NE + V (w/o parsing)	26	42	32	28	40	33
H + NE + V (with parsing)	38	49	43	40	45	42

Table 6: Accuracy of string slots with and without full parsing

on the official web site). The same history file downloaded from the official web site is uniformly used for scoring the output templates of all systems (the history file contains the arbitration decisions for ambiguous cases).

We conducted statistical significance test, using the approximate randomization method adopted in MUC-4. Table 5 shows the systems that are not significantly different from Alice-ME.

Our system Alice-ME, using a learning approach, is able to achieve accuracy competitive to the best of the MUC-4 participating systems, which were all built using manually engineered rules. We also observed that ME and SVM, the more recent machine learning algorithms, performed better than DT and NB.

Full Parsing. To illustrate the benefit of full parsing, we conducted experiments using a subset of features, with and without full parsing. We used ME as the learning algorithm in these experiments. The results on string slots are summarized in Table 6. The baseline system used only two features, head word (H) and named entity class (NE). Next, we added three features, VAg, VPa, and V-Prep. Without full parsing, these verbs were obtained based on the immediately preceding (or following) verb of a noun phrase, and the voice of the verb. With full parsing, these verbs were obtained based on traversing the full parse tree. The results indicate that verb features contribute to the performance of the system, even without full parsing. With full parsing, verbs can be determined more accurately, leading to better overall performance.

5 Discussion

Although the best MUC-4 participating systems, GE/GE-CMU, still outperform Alice-ME, it must be noted that for GE, “10 1/2 person months” were spent on MUC-4 using the GE NLTOOLSET , after spending “15 person months” on MUC-3 (Rau et al., 1992). With a learning approach, IE systems are more portable across domains.

Not all occurrences of a string in a document that match a slot fill of a template provide good positive training examples. For example, in the same document, there might be the following sentences “THE MNR REPORTS THE KIDNAPPING OF OQUELI COLINDRES...”, followed by “OQUELI COLINDRES ARRIVED IN GUATEMALA ON 11 JANUARY”. In this case, only the first occurrence of OQUELI COLINDRES should be used as a positive example for the human target slot. However, ALICE does not have access to such information, since the MUC-4 training documents are not annotated (i.e., only templates are provided, but the text strings in a document are not marked). Thus, ALICE currently uses all occurrences of "OQUELI COLINDRES" as positive training examples, which introduces noise in the training data. We believe that annotating the string occurrences in training documents will provide higher quality training data for the learning approach and hence further improve accuracy.

Although part-of-speech taggers often boast of accuracy over 95%, the errors they make can be fatal to the parsing of sentences. For example, they often tend to confuse "VBN" with "VBD", which could change the entire parse tree. The MUC-4 corpus was provided as uppercase text, and this also has a negative impact on the named entity recognizer and part-of-speech tagger, which both make use of case information.

Learning approaches have been shown to perform on par or even outperform knowledge-engineering approaches in many NLP tasks. However, the full-scale scenario template IE task was still dominated by knowledge-engineering approaches. In this paper, we demonstrate that using *both* state-of-art learning algorithms and full parsing, learning approaches can rival knowledge-engineering ones, bringing us a step closer to building full-scale IE systems in a domain-independent fashion with state-of-the-art accuracy.

Acknowledgements

We thank Kian Ming Adam Chai for the implementation of the full parser.

References

M. E. Califf and R. J. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI99*, pages 328–334.

E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. 1993. Equations for part-of-speech tagging. In *Proceedings of AAAI93*, pages 784–789.

H. L. Chieu and H. T. Ng. 2002a. A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of AAAI02*, pages 786–791.

H. L. Chieu and H. T. Ng. 2002b. Named entity recognition: A maximum entropy approach using global information. In *Proceedings of COLING02*, pages 190–196.

F. Ciravegna. 2001. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of IJCAI01*, pages 1251–1256.

M. Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

R. O. Duda and P. E. Hart. 1973. *Pattern Classification and Scene Analysis*. Wiley, New York.

D. Fisher, S. Soderland, J. McCarthy, F. Feng, and W. Lehnert. 1995. Description of the UMass system as used for MUC-6. In *Proceedings of MUC-6*, pages 127–140.

D. Gildea and D. Jurafsky. 2000. Automatic labelling of semantic roles. In *Proceedings of ACL00*, pages 512–520.

S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, R. Weischedel, and the Annotation Group. 1998. Algorithms that learn to extract information BBN: Description of the SIFT system as used for MUC-7. In *Proceedings of MUC-7*.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco.

A. Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

L. Rau, G. Krupka, and P. Jacobs. 1992. GE NL-TOOLSET: MUC-4 test results and analysis. In *Proceedings of MUC-4*, pages 94–99.

D. Roth and W. Yih. 2001. Relational learning via propositional algorithms: An information extraction case study. In *Proceedings of IJCAI01*, pages 1257–1263.

S. Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1/2/3):233–272.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.