

Information extraction by embedding HMM to the induced linguistic feature space

Hyun Chul Lee¹ and Jian Chang Mao²

¹ Department of Computer Science
University of Toronto
Toronto, Ontario, M5S 3G4
leehyun@cs.toronto.edu

² Verity, Inc.
892 Ross Drive
Sunnyvale, California
jmoo@verity.com

Abstract. In recent years, several research efforts have been devoted to the automatization of information extraction (IE) from text. While most of works have been based on the inductive learning of extraction patterns, Hidden Markov Models (HMMs) have shown to be a powerful alternative. In the traditional HMMs, the observation probabilities are usually represented as a multinomial distribution over the vocabulary of words. In this paper, we propose and evaluate an approach for IE by representing the extracted grammatical patterns as states of a HMM. Our approach first learns some simple information extraction rules and then trains a HMM with these extracted rules. Our experiments suggest that with the incorporation of simple extraction rules, the reliability and the performance of HMM based IE system are greatly enhanced.

1 Introduction to the information extraction problem

The main objective of information extraction (IE) from text is to extract corresponding relevant data for some entities or fields from a corpus of documents. Usually, the relevant data to be extracted are pre-specified by the user before the extraction task. Because of the corpus size that has to be handled especially for applications like populating databases, automatic classification of documents and identifying significant but unknown relationships among objects, an automated information extraction method is normally desirable. Being this the case, there have been several works [1–10] that applied various machine learning techniques to IE.

These techniques fall into one of two categories, namely *extraction pattern learning methods* and *probabilistic methods*. Various IE systems like Nymble, SRV, Rapier, WHISK, (LP)², SNoW-IE are *extraction pattern learning methods*. These methods aim to learn extraction patterns or rules from the labeled training data by generalizing contexts of occurrence of relevant entities. The interested reader can consult [11] which provides an excellent survey of the field.

Meanwhile, *probabilistic methods* aim to construct an implicit probabilistic model of the corpus for IE task. Most of the existing probabilistic methods are based on Hidden Markov Models (HMMs). In a HMM, each labeled training document is represented as a sequence of states emitting words from the document corpus. Extracting the corresponding entities from a new document is equivalent to finding the most probable state sequence that had produced that document. Early HMM approaches [1, 3, 5] completely neglected the fact that the data that they are dealing with are normally complex natural language texts since each document was simply represented as a sequence of tokens. Successive approaches [7, 10] made one step further by incorporating into their models some grammatical structure regularities of sentences. However, none of the existing probabilistic approaches does fully explore the wide array of linguistic information as the extraction pattern learning methods aim to do. Furthermore, like in many machine learning techniques, constructing an appropriate probabilistic model for IE task, which is expressive enough, is a challenging task. Complex models might result in poor parameter estimation because training data are often highly sparse while excessively simple models which will result in robust parameter estimates might not be powerful enough to capture the complexity of the task. In spite of some works [3, 5] in this direction, no satisfactory solution has been found yet.

In this paper, we propose a novel HMM-based IE method as an alternative to the traditional HMM-based IE systems. In our HMM, a document is represented as a sequence of extracted patterns obtained by a simple inductive algorithm rather than as a sequence of simple tokens. To the best of our knowledge, our approach is the first attempt to integrate techniques used by extraction pattern learning methods into a probabilistic method. Moreover, our method is simple and fast making it particularly suitable for many real applications.

The rest of this paper is organized as follows. Section 2 introduces the motivation behind our approach. Section 3 describes our model in detail. Section 4 shows the experimental studies that we conducted. Finally, section 5 presents our conclusions.

2 Motivation

We start describing the motivation behind our approach. Consider the task of extracting the purchasing prices from the collection of documents on the corporation acquisition activities. Two typical sentences containing these purchasing prices would look like

- | |
|---|
| a) The purchasing price of the company was <u>£27 million</u> . |
| b) The initial offer for the company is <u>\$1.5 billion</u> . |

where £27 million and \$1.5 billion correspond to entities that we want to extract. We suppose that the sentence a) is in the training corpus while the sentence b) is one from which the entities are going to be extracted. By analyzing these sentences we are able to deduce some simple observations that are applicable

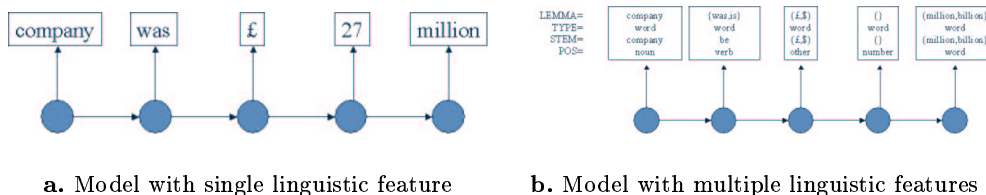


Fig. 1. Document Representation

to the IE task in general: (1) a particular set of tokens or delimiters indicate the potential presence of particular information. In our sample sentences, for instance, words like “price (offer)” and “£(\$)” along with “company” strongly signal the potential presence of purchasing price entity. Therefore, the model for IE would be flexible enough to properly handle this issue. (2) while two sentences might be different in *content*, they can be similar in *structure*. In our sample sentences, both sentences contain the word “company” followed by “was” and “is”, both having “be” as stem word, and finally terminated with “£27 million” and “\$1.5 billion” which are morphologically similar.

While the traditional probabilistic methods have explored (1), none of the previous probabilistic methods have properly addressed (2). Thus, a sentence representation that emphasizes morphological structures as Figure 1b would be more natural than the simple token based representation as that of Figure 1a. In our particular example, sentence b) would hardly be captured by the traditional representation because it was never seen in the training while it will be able to be captured by our proposed representation. Therefore, in this paper, we propose a model where each sentence is viewed as the consequence of a stochastic process with various types of states, each one emitting multiple linguistic features learned from the training data instead of simple tokens.

3 Embeddiment of HMM to the space of induced linguistic features

3.1 Preliminaries

Certainly, there is no limit on the amount of possible grammatical information that we can use for our model. However, we do not rely on lot of linguistic information for representing our candidate instances as some inductive learning methods do. Even with this limited linguistic information, our system shows to be powerful as shown through our experimental results. A shallow parser *TestTok*, developed at *Verity Inc.*, is used for this purpose. Provided a text as input, *TestTok* produces simple word type and part-of-speech tags for each word of the text in addition to its length, stem form and position in the text. We do not use

all information produced by *TestTok* but we only keep word type, stem form, part-of-speech (POS) tags.

Consequently, each word of the candidate instance is represented as a vector that consists of lemma, type, stem, and POS fields and we call this vector representation as token, i.e. a token h is a feature vector of the form $(h(\text{lemma}), h(\text{type}), h(\text{stem}), h(\text{pos}), h(\text{label}))$. Let *instance* be an ordered bag of single valued tokens that consists of n tokens. Therefore, two sample sentences of the previous section are examples of instances. When the order in an instance g is ignored, the instance is simply denoted by \tilde{g} . Let the rule be the union of more than one instance of the same size. Even though, an instance is a particular case of the rule, most of times, by rules we refer the bag of multi-valued tokens. For instance, $\{(\text{company, word, company, noun}), (\{\text{was, is}\}, \text{word, be, verb}), (\{\cdot\}, \text{word, }\{\cdot\}, \text{other}), (\{\cdot\}, \text{word, }\{\cdot\}, \text{number})\}, (\{\text{million, billion}\}, \text{word, }\{\text{million, billion}\}, \text{word})$ is a rule. We say that a *rule* r covers the *instance* g if for each $r_i, g_i \subset r_i$ is held. Later in the section, we explain how our rules are constructed. When the order in a rule r is neglected, it is simply denoted by \tilde{r} .

3.2 Overall Description

The main difference between our model and the traditional HMM is in the pre-processing of documents in order to induce linguistic features, which in turn, will be used as input to the HMM. Basic strategies for training and extraction, thus, can be summarized as follows:

– **Training**

1. For each document in the training data, identify and extract those candidate instances from the document which contain the entity segments that we want to extract.
2. Induce a set of linguistic features with the previously obtained candidate fragments.
3. Using the induced linguistic features as the symbol set, perform the training of HMM.

– **Extraction**

1. Given a new document, map each token of the document to the space of already induced linguistic features.
2. With this new representation of document, find the most probable state sequence in HMM.
3. Extract the entity segments according to the previously obtained state sequence.

Next, we explain in detail each step of the training and the extraction.

3.3 Feature induction using sequential covering algorithm

Once the candidate instances have been identified and extracted from the training data, the next step is to induce linguistic features for our model by constructing rules from these instances. For this purpose, we first define two notions: the similarity between rules and the unification operator.

Similarity Measure: When a sentence is represented by the sequence of linguistic features, one question that we have to deal with is which linguistic features would be suitable for this representation. For example, we need to determine how appropriate is to use the rule $\{(company, word, company, noun), (\{was, is\}, word, be, verb), (\{\cdot\}, word, \{\cdot\}, other), (\{\cdot\}, word, \{\cdot\}, number)\}, (\{million, billion\}, word, \{million, billion\}, word)$ to represent the instance $\{company, was, ?, 27, million\}$. Therefore, we need to define the similarity measure between rules. We define a similarity measure which is similar in spirit to the traditional euclidean l_1 norm. Additionally, we allow any entry of the feature vector to hold the “universal” value which will be denoted as “.”. That is, a field having the “universal” value will be able to hold any possible value for that particular field. In this way, we are restricting the number of possible values for an entry of feature vector. More formally, given two rules x and y , the similarity measure between x and y , denoted by $sim(x, y)$, is defined as

$$sim(x, y) = \begin{cases} \frac{\gamma(\sum_{i=1}^n y_i)}{\infty} & \text{if } x \text{ and } y \text{ are of the same size} \\ \infty & \text{otherwise} \end{cases}$$

where

$$\gamma(x_i, y_i) = \sum_{s \in Q} \lambda_s \delta(x_i(s), y_i(s))$$

$$Q = \{lemma, type, word, POS\}$$

with

$$\delta(x_i(s), y_i(s)) = \begin{cases} \beta_1 & \text{if } x_i(s) = y_i(s) \text{ and } x_i(s), y_i(s) \neq \cdot \\ \beta_2 & \text{if } x_i(s) \subset y_i(s) \text{ or } x_i(s) \supset y_i(s) \\ \beta_3 & \text{if } x_i(s) = \cdot \text{ or } y_i(s) = \cdot \\ \beta_4 & \text{otherwise} \end{cases}$$

such that

$$\sum_{s \in Q} \lambda_s = \beta_1 \text{ and } \beta_1 \geq \beta_2 \geq \beta_3 \geq \beta_4$$

Example 1. Suppose that values for β_i 's, λ_{token} , λ_{pos} , λ_{stem} and λ_{type} are specified as

$$\begin{aligned} - \beta_1 &= 1, \beta_2 = \frac{2}{3}, \beta_3 = \frac{1}{3}, \beta_4 = 0 \\ - \lambda_{token} &= \frac{3}{5}, \lambda_{pos} = \frac{18}{5}, \lambda_{stem} = \frac{9}{5}, \lambda_{type} = 3 \end{aligned}$$

and x, y are given as

$$x = \left\{ \begin{array}{l} x_1 = (company, word, company, noun) \\ x_2 = (\{was, is\}, word, be, verb) \\ x_3 = (6, word, \cdot, number) \end{array} \right\}$$

$$y = \left\{ \begin{array}{l} y_1 = (\cdot, word, \cdot, noun) \\ y_2 = (is, word, be, verb) \\ y_3 = (5, word, 5, number) \end{array} \right\}$$

then one can easily verify that $\delta(x_1, y_1)=0.4178, \delta(x_2, y_2)=0.4179, \delta(x_3, y_3)=0.41777$.

Note that parameters λ_s determine the importance of each field in the feature vector representation.

Rule induction: Each time two rules x and y are unified into a new rule, the new rule can be generalized to cover more number of instances. Thus, we need to define an operation that dictates how two rules should be combined and then generalized. While some complex methods are used by most of extraction pattern learning methods [11], we use a simple generalization approach. Each time we observe a new instance, we unify each feature of the current rule with the corresponding feature of the new instance until the maximum number of elements, which is controlled by a threshold value, for each entry is reached. More precisely, the unification operation, $unif(x, y)$ is defined as:

$$unif(x, y) = \{\gamma(x_1, y_1), \dots, \gamma(x_n, y_n)\}$$

where $Q = \{lemma, type, word, POS\}$ and $\gamma(x_i, y_i)(s)$ for any $s \in Q$ is defined as

$$\gamma(x_i, y_i)(s) = \begin{cases} \cdot & \text{if } |x_i(s)| \geq tsize \text{ or } |y_i(s)| \geq tsize \\ y_i(s) & \text{if } x_i(s) \subseteq y_i(s) \\ x_i(s) & \text{if } y_i(s) \subseteq x_i(s) \end{cases}$$

The parameter $tsize$, which determines the level of generalization, is specified by the user. For instance, $tsize = 3$ means that whenever the number of elements in one of the entries of either x or y is greater than 3, the value of the corresponding entry will be replaced with the universal value, “.”.

Having defined the notion of similarity between rules and unification operator, now we describe the sequential covering algorithm used for the linguistic feature induction step. Even though, in theory, any extraction pattern learning method can be used for the the feature induction step, we determined that a simple *sequential covering algorithm*, which motivated by *Crystal* [12], would be sufficient for our purpose due to its simplicity. A sequential covering algorithm is an algorithm which greedily finds rules that apply to cover instances. Each time a rule is generalized, the instances that are covered by the rule are eliminated from the search space before a new rule is searched for. Particularly, our sequential covering algorithm accepts as input the set of instances $I = \{i_1, i_2, \dots, i_k\}$ and outputs a set of linguistic feature vectors produced as the union of induced rules. From now on, we refer to this set of feature vectors as the *induced linguistic feature space*. The detailed description of algorithm is given as Figure 2 where *matches* denotes the total number of times that the rule u covers candidate instances and *wrong* denotes the number of times that the negative examples are covered by u . Constant δ is used to determine the degree of rule generalization. Even though, a small value of δ ³ is normally used when this type of covering algorithm is directly applied to the IE, for our linguistic feature induction purpose, this is not necessarily the case.

³ For *Crystal*, for instance, 0.1 is employed

<p>Algorithm</p> <p>Input: I: set of all instances</p> <p>Output: W: set of induced linguistic feature vectors</p> <p>Parameters: δ: error threshold <i>tsize</i>: level of generalization</p> <p>Procedure:</p> <p>[1] Compute $sim(i_p, i_q), \forall i_p, i_q$</p> <p>[2] Let $R = \emptyset, W = \emptyset$</p> <p>[3] Do for each $i_l \in I$ not covered by R</p> <p>[4] Let $u = i_l, error(u) = 0$</p> <p>[5] Do while $error(u) \leq \delta$</p> <p>[6] Let $r = u$</p> <p>[7] Let i_p the most similar instance to r of the same target size</p> <p>[8] If $i_p = \emptyset$ then exit</p> <p>[9] $u = unif(i_p, r)$</p> <p>[10] Test u on the training data</p> <p>[11] Let $error(u) = \frac{wrong}{matches}$</p> <p>[12] add r to R</p> <p>[13] For each $r \in R$</p> <p>[14] $W = W \cup r_i$</p> <p>[15] return W</p>
--

Fig. 2. Sequential Covering Algorithm

3.4 eHMM

With the set of induced linguistic features produced by our sequential covering algorithm, we train our HMM. For the reference purpose, from now on we call our model as eHMM.

Not many topologies have been proposed for HMMs for IE. But, we found the topology proposed by [3] as particularly suitable for our task. Its topology consists of background, prefix, target and suffix states as shown in Figure 3. In addition to the characteristics of the original topology [3], our model automatically learns from the training data the number of parallel length-differentiated target paths. More precisely, with the target size of each instance, we produce an ordered list of target sizes. The biggest target size of the first consequent sequence found in the list is taken as the number of length-differentiated target paths of HMM. For example, if we have obtained target sizes $\{1, 2, 3, 4, 5, 9, 13\}$, then 5 will be the number of parallel length-differentiated target paths. Under this particular HMM topology, and a non-empty document, only a single and

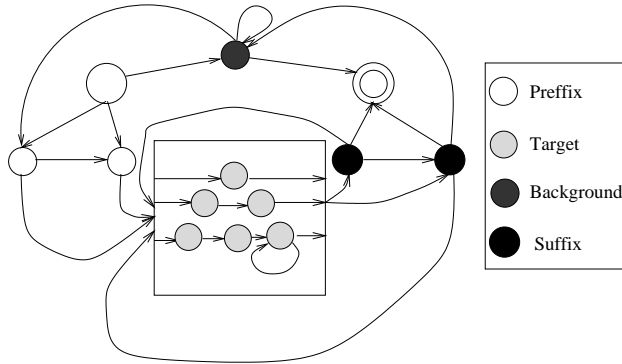


Fig. 3. A typical hmm topology with prefix/suffix size=2, target size=2, number of parallel length-differentiated target paths=2

unambiguous path is possible. Thus, the transition probabilities between states are easily computed by the standard maximum likelihood with ratios of count.

Next, we describe the calculation of the emission probabilities. We first define the map between the vocabulary set of words and the induced linguistic feature space as follows. Given token c_i , let $E(c_i)$ be the most similar element to c_i in the induced linguistic feature space. If we denote by D the collection of training documents, then the emission probability of symbol $b \in W$ from state s , $P(b|s)$, is given by

$$P(b|s) = \sum_{d \in D} P(b|d, s)P(d|s) = \sum_{d \in D} P(b|d, s)P(s|d) \frac{P(d)}{P(s)}$$

Note that $P(s|d)$, $P(d)$ and $P(s)$ are easily estimated. Thus, we only deal with $P(b|d, s)$. One simple way of estimating $P(b|d, s)$ might be by counting the number of times that c_i such that $E(c_i) = b$ is encountered in the document d as the state s . We refer this value as $N(b, s, d)$. This simple approach might not work well due to the sparseness of training data. Instead, we propose a smoothing method that distributes weights among all possible observations according to their similarity to the token being estimated: whereas with the standard approach $N(b, s, d)$ would be incremented by one, each time that c_i such that $E(c_i) = b$ is observed in the training set, with our smoothing method, each time the word c_i is observed in the document d , we associate with this observation the probability of observing each induced linguistic feature. This probability would depend on the closeness of the considered linguistic feature to b . Obviously, we have to be careful since we want to have this estimation result sufficiently discriminative. Therefore, all $N(b, s, d)$ s are simultaneously incremented by $e(b, s, d)$

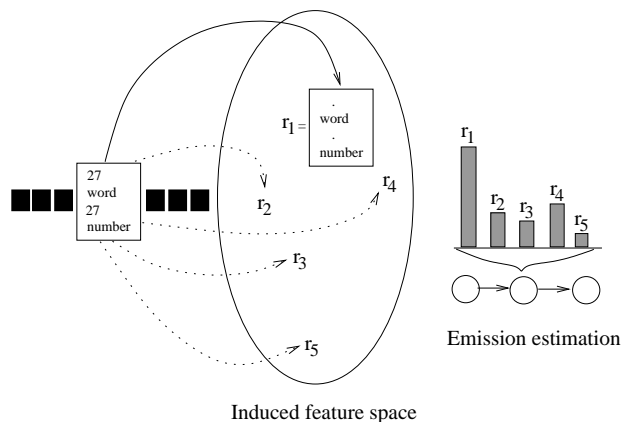


Fig. 4. Parameter Estimation

whenever b is observed as illustrated in Figure 4. $e(b, s, d)$ will be defined as

$$e(b, s, d) = \begin{cases} 1 & \text{if } b = \arg \max_{w \in W} \text{sim}(w, c_i) \text{ and } c_i \text{ is labeled as } s \\ \alpha * \text{sim}(b, w_i) & \text{if } b \neq \arg \max_{w \in W} \text{sim}(w, c_i) \text{ and } c_i \text{ is labeled as } s \end{cases}$$

where $0 \leq \alpha \leq 1$ denotes the degree of smoothing. Choosing excessively high value of α would make our estimation flat. We usually use values ranging from 0.00001 to 0.000001. Finding the optimal value for α , though, requires separate research and it is behind to the scope of our work.

3.5 Extraction

It is known that a naive approach to find the most likely state sequence given a HMM and a sequence of symbols takes an exponential time in the sequence length. An alternative dynamic programming solution, *Viterbi algorithm*, which has the complexity $O(TN^2)$ where T =length of the sequence and N =the number of states in the model, is normally used [13]. For the extraction step, we employ the standard Viterbi algorithm without any modification. Given new document represented as the sequence of tokens $\{c_1, c_2, \dots, c_{n-1}, c_n\}$, we map this sequence to the induced linguistic feature space to produce a new sequence $\{E(c_1), E(c_2), \dots, E(c_{n-1}), E(c_n)\}$. Next, using the Viterbi algorithm, we obtain the most probable state sequence in order to extract the corresponding entities from the document.

4 Experimental Results

Our approach was tested on two corpora from informal domains: the CMU seminar announcements corpus and the corpus that is composed of abstracts gathered

System	stime			etime			location			speaker		
	Rec.	Prec.	F-m.	Rec.	Prec.	F-m.	Rec.	Prec.	F-m.	Rec.	Prec.	F-m.
eHMM	96.1	95.1	95.9	94.2	96.5	95.4	83.7	94.2	88.6	71.2	69.4	70.3
(LP) ²	99.0	99.0	99.0	97.0	94.0	95.4	66.0	87.0	75.0	70.0	87.0	77.5
HMM-shrinkage	-	-	99.1	-	-	59.5	-	-	83.9	-	-	71.1
HMM	-	-	99.1	-	-	81.4	-	-	73.5	-	-	51.3
Rapier	95.3	96.5	95.9	96.6	96.8	96.7	61.5	91.0	73.4	39.4	80.9	52.9
SRV	98.4	98.6	98.5	92.6	67.3	77.9	70.1	74.5	72.2	58.4	54.4	56.2
WHISK	1	86.2	92.5	87.2	85.0	86.0	55.1	83.6	66.4	11.1	52.6	18.3
SNoW-IE	99.6	99.6	99.6	95.0	97.6	96.3	64.1	90.9	75.2	66.3	83.3	73.8
NB-IE	98.3	98.3	98.3	92.8	96.5	94.6	62.0	76.8	68.6	32.0	40.3	35.7

Table 1. Performance Results of different IE systems

from the MEDLINE database. CMU seminar announcement corpus has been investigated by various researchers. We have chosen CMU seminar announcements corpus for the purpose of comparison. We also tested our approach on MEDLINE datasets in order to study the effect of each field of the feature vector on the overall performance. For both experiments, the performance is calculated in terms of standard measures like precision, defined as the fraction of correct tuples among those tuples that are extracted by the model and recall, defined as the fraction of correct tuples extracted by the model over the total number of true tuples that exist in the dataset. F-measure, the harmonic average of recall and precision, is also used as auxiliary information.

The CMU seminar announcements corpus consists of 485 documents. Each announcement contains some tags for target slots. The task consists of uniquely identifying speaker name, starting time, ending time and location of each seminar. The corpus contained 757 *speakers*, 644 *locations*, 982 *stime* and 433 *etime*. *etime*, *speaker* and *location* are missing from 48%, 16%, 5% of documents correspondingly making the dataset sparse especially for *etime*. We report results using the same validation test as other publications concerning this dataset [8, 2]. The reported results are based on 50/50 split of the corpus, i.e. the randomly chosen 50% of the corpus was used for training while the rest was used for testing. We employed $\delta = 0.3$ and $tsize = 3$ for the sequential covering algorithm. Moreover, *part-of-speech* and *type* fields are more emphasized than other fields. The reported figures are averaged over five runs. Table 1 compares the performance of our system with that of several previous attempts at the CMU seminar corpus.

Our system performs comparably to the best systems in each category, while clearly outperforming all other systems in finding *location*. *location* entity is particularly benefited by the usage of extraction patterns instead of single tokens for its representation since many location entities are found in the form of a place name followed by a place number. One may also note that the eHMM does not show the same drawback as that of the HMM with shrinkage. As pointed out in [3], “shrinkage” can sometimes hurt the performance as sparsely trained states tend to emit tokens they have never seen in training. In CMU corpus,

eHMM	Disease	Gene	Location	Protein
lemma	44.99	22.53	49.80	30.03
type	37.87	22.29	46.40	29.67
stem	47.79	22.45	50.08	27.93
POS	36.34	21.58	48.21	28.78

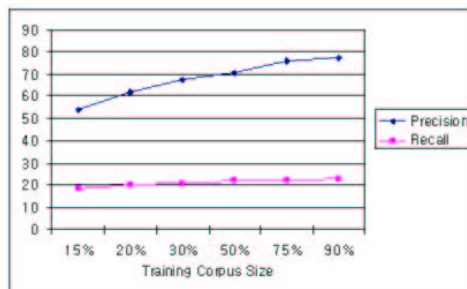


Table 2. F-measure of eHMM emphasizing different linguistic fields **Fig. 5.** Learning curve for Precision and Recall

for example, the sparsity of ending time entity has some negative effect on the performance of HMM when “shrinkage” is used. In contrast, our eHMM shows a consistent performance over all entities including *etime* showing that our smoothing method performs well even when the training set is sparse.

Next, we tested our approach on two datasets, which are composed of abstracts gathered from the MEDLINE database [14]. These datasets are originally used for the IE task of binary relations in [7, 10], but we use them to individually extract each entity. The task was much more challenging since the corpus was much sparser than the CMU seminar announcement corpus. The first set of MEDLINE dataset consists of *location* and *protein* entities being composed of 769 positive and 6360 negative sentences. The positive sentences contain 950 *location* and 780 *protein* entities. The second set of MEDLINE dataset consists of *disease* and *gene* entities. It has 829 positive and 11771 negative sentences. The positive sentences represent 647 *disease* and 856 *gene* entities respectively.

Our experiment with these datasets showed that emphasis of different field (lemma, type, stem, word) of the feature vector does not affect the overall performance of eHMM as illustrated in Table 2. Moreover, increasing the size of the training corpus does not dramatically improve the performance in terms of precision and recall, as illustrated in Figure 5 which presents the learning curve—precision and recall average over all fields, as a function of training sample size. Since eHMM is based on extraction patterns, small number of training samples were sufficient resulting in consistent recall, across different training sample sizes. As the training sample size increases, more accurate extraction patterns were generated resulting in the increase of precision.

5 Conclusion

We have proposed an approach to learning models for information extraction by combining the previous probabilistic and inductive pattern learning methods. We employ a shallow parser to obtain some grammatical information for sentences which are used to induce some simple rules. These simple rules, in

turn, are used to construct the symbol set for the HMM. Our experimental results strongly indicate that with our approach we are able to enhance traditional hmm based information extraction systems. Our work suggest many interesting and challenging directions for future work. Above all, a natural extension of our work is to find a more complete way of combining probabilistic and extraction pattern learning methods into one single model.

References

1. Bikel, D., Miller, S., Schwartz, R., Weischedel, R.: Nymble: a high performance learning name finder. In: Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington, D.C., Morgan Kaufman (1997) 194–201
2. Ciravegna, F.: Adaptive information extraction from text by rule induction and generalization. In: Proceedings of the seventeenth international joint conference on artificial intelligence, Washington, D.C., Morgan Kaufmann (2001)
3. Freitag, D., McCallum, A.: Information extraction using hmms and shrinkage. In: Proceedings of AAAI-99 Workshop on machine learning for information extraction, Menlo Park, California, AAAI (1999) 31–36
4. Freitag, D.: Machine learning for information extraction in informal domains. *Machine Learning* **39** (2000) 169–202
5. Freitag, D., McCallum, A.: Information extraction with hmm structures learned by stochastic optimization. In: Proceedings of the seventeenth national conference on artificial intelligence, Austin, Texas, AAAI (2000) 584–589
6. Peshkin, L., Pfeffer, A.: Bayesian information extraction network. In: Proceedings of the eighteenth international joint conference on artificial intelligence, Acapulco, Mexico, Morgan Kaufman (2003)
7. Ray, S., Craven, M.: Representing sentence structure in hidden markov models for information extraction. In: Proceedings of the seventeenth international joint conference on artificial intelligence, Seattle, Washington, Morgan Kaufman (2001) 1273–1279
8. Roth, D., Yih, W.: Relational learning via propositional algorithms: an information extraction case study. In: Proceedings of the seventeenth international joint conference on artificial intelligence, Seattle, Washington, Morgan Kaufman (2001)
9. Seymore, K., McCallum, A., Rosenfeld, R.: Learning hidden markov model structure for information extraction. In: Proceedings of AAAI-99 Workshop on machine learning for information extraction, Menlo Park, California, AAAI (1999) 37–42
10. Skounakis, M., Craven, M., Ray, S.: Hierarchical hidden markov models for information extraction. In: Proceedings of the eighteenth international joint conference on artificial intelligence, Acapulco, Mexico, Morgan Kaufman (2003)
11. Muslea, I.: Extraction patterns for information extraction tasks: a survey. In: Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction, Menlo Park, California, AAAI (1999) 1–6
12. Soderland, S., Fisher, D., Aseltine, J., Lehnert, W.: Crystal: inducing a conceptual dictionary. In: Proceedings of the fourteenth international joint conference on artificial intelligence, Montreal, Canada, Morgan Kaufmann (1995) 1314–1319
13. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**(2) (1989) 257–285
14. National Library of Medicine, ed.: The MEDLINE database. NIH, <http://www.ncbi.nlm.nih.gov/PubMed/> (2003)