Role identification from free text using hidden Markov models

Georgios Sigletos, Georgios Paliouras and Vangelis Karkaletsis

Software and Knowledge Engineering Laboratory
Institute of Informatics and Telecommunications,
N.C.S.R. "Demokritos",
Tel: +301-6503197, Fax: +301-6532175
E-mail: {sigletos, paliourg, vangelis}@iit.demokritos.gr

Abstract. In this paper we explore the use of hidden Markov models on the task of role identification from free text. Role identification is an important stage of the information extraction process, assigning roles to particular types of entities with respect to a particular event. Hidden Markov models (HMMs) have been shown to achieve good performance when applied to information extraction tasks in both semistructured and free text. The main contribution of this work is the analysis of whether and how linguistic processing of textual data can improve the extraction performance of HMMs. The emphasis is on the minimal use of computationally expensive linguistic analysis. The overall conclusion is that the performance of HMMs is still worse than an equivalent manually constructed system. However, clear paths for improvement of the method are shown, aiming at a method, which is easily adaptable to new domains.

1. Introduction

Role identification is the subtask of *information extraction*, dealing with the assignment of event-specific roles to the various entities mentioned in a piece of text that describes an event. In the information extraction process, as defined in the Message Understanding Conferences [8], role identification is part of the *scenario template-filling* task, which is the ultimate goal of the information extraction process. Thus, role identification is a hard task, often requiring significant use of computationally expensive linguistic processing methods.

In this paper we investigate the problem of role identification using hidden Markov models (HMMs). Hidden Markov modeling is a powerful statistical learning technique with widespread application, mostly in the area of speech recognition [11]. HMMs have also been applied successfully to other language related tasks, including part-of-speech tagging [2], named entity recognition [1] and text segmentation [15]. The main advantage of HMMs in language modeling is the fact that they are well suited for the modeling of sequential data, such as spoken or written language. Another serious motivation behind the use of HMMs for text-based tasks is their strong statistical foundations, which provide a sound theoretical basis for the constructed models. On the other hand, an important concern with the use of HMMs

is the large amount of training data required to acquire good estimates of the model parameters.

Recent research has suggested the use of HMMs for the task of role identification from either semistructured or free text. Leek in [7] designed HMMs to extract gene locations from biomedical texts. Freitag & McCallum in [3] and [4], used HMMs for information extraction both from newsgroups and a collection of Reuter's articles. The focus of that work was on techniques that either improve the estimation of model parameters [3] or learn the model structure from training data [4]. However, the use of HMMs for role identification from free text is largely unexplored territory and there are many important issues to be examined.

In this paper we examine for the first time the use HMMs for role identification from Greek texts. For this purpose, we have used a collection of Greek financial articles describing company acquisitions, which was used in the MITOS R&D project [5]. Unlike previous work on HMMs for role identification, we pay particular attention to whether and how linguistic processing of textual data can improve the extraction performance of HMMs. This is a difficult issue, because the initial intuition that linguistic analysis is likely to help in extracting information from natural language, has to face the reality of high computationally costs for using linguistic analysis tools. Therefore, it is important to identify the minimum necessary linguistic processing for improving the performance of information extraction, while maintaining the computational efficiency of the process. Along this line of thought, we performed various types of linguistic preprocessing to our dataset, and considered different data representations, where linguistic information was represented as part of the text in a *sequential* form. The motivation for the sequential representation is the suitability of HMMs for modeling sequential data.

The rest of this paper is structured as follows. In section 2 we review the basic theory of HMMs and discuss how HMMs can be used for role identification. In section 3, we present experimental results on our dataset varying the use of linguistic processing. Finally, we conclude in section 5, discussing potential improvements of the method.

2. HMMs for Role Identification

2.1 Basic Theory

A hidden Markov model is an extension of a Markov process where the observation is a probabilistic function of a state. The elements that characterize an HMM are:

- A set of N individual states $S = \{s_1, s_2, \ldots, s_N\}$, often interconnected in a way that any state can be reached from any other state (*ergodic* model).
- A discrete vocabulary of M observation symbols $V = \{v_1, v_2, \ldots, v_M\}$.
- An NxN state transition probability matrix $A = \{a_{ij}\}$, indicating the probability of transitioning from state i to state j. Here we deal with *first-order* HMMs, which

means that transitioning to the next state *j* at time *t+1* depends only on the current state *i* at time *t*, i.e., $P[s_j(t+1) \mid s_i(t)\, s_k(t-1)\ldots] = P[s_j(t+1) \mid s_i(t)] = a_{ij}$.

- An *N*x*M* observation symbol probability matrix $B = \{b_j(k)\}$ indicating the probability of observing symbol v_k at state s_j.
- An *N*x*1* initial state matrix $\pi = \{\pi_i\} = \{P[s_i(1)]\}$, indicating the probability of being at state s_i at time *t=1*.

An HMM is a probabilistic generative model, whereby a sequence of symbols, denoted as $O = \{o_1 o_2 \ldots o_T\}$, is produced by starting from an initial state *i* (with probability π_i), emitting an output symbol $v_k = o_1$ (with probability $b_i(k)$), transitioning to a new state *j* (with probability a_{ij}) emitting a new symbol and so on until reaching the final state at time T and emitting the output symbol o_T. Here we also deal with *discrete output* HMMs, meaning that *O* is a sequence of discrete symbols, chosen from the vocabulary *V*.

The three classic issues with HMMs are the following [12]:

1. Given the parameters $\lambda = (A, B, \pi)$ of an HMM and a sequence of symbols, how can we efficiently compute the probability $P(O \mid \lambda)$, that the observation sequence was produced by the HMM? This is an *evaluation* problem, which allows us to choose the model which best matches the sequence.
2. Given the parameters $\lambda = (A, B, \pi)$ of an HMM and a sequence of symbols, how can we efficiently compute the most likely state sequence $Q=\{q_1 q_2 \ldots q_T\}$ associated with the symbol sequence? The state sequence *Q* is *hidden* and can be *observed* only through the sequence *O*. This issue relates to the "uncovering" of the hidden state sequence.
3. How can we efficiently estimate the parameters $\lambda = (A, B, \pi)$ to maximize $P(O \mid \lambda)$? This is the most difficult of the three problems, dealing with the *training* of an HMM given a set of observation sequences.

The above three problems can be solved using the *Forward-Backward*, *Viterbi* and *Baum-Welch* algorithms respectively, as described in [12].

A key insight into the use of HMMs for language related tasks is that state transitions are modeled by a *bigram* model emitting label types from a *N*-length discrete vocabulary (just as with traditional Markov models), while each state is a label-specific *unigram* language model, emitting tokens from a *M*-length discrete vocabulary.

2.2 Using HMMs for role identification

In order to train HMMs for the role identification task, we make the following assumptions, inspired by related work in [3] and [4].

- An HMM models the entire document, thus not requiring any segmentation of the document into sentences or other pieces of text. Each training document is modeled as a sequence *O*, of lexical units (tokens). The discrete tokens of all the training sequences constitute the discrete alphabet *V* of the HMM.
- A separate HMM is constructed for each role of the event. In this paper we deal with a collection of Greek articles describing company acquisitions. For this event, we are interested in four roles: the *buyer* company, the company that is *acquired*, the acquisition *rate* and the acquisition *amount*. Thus, we construct four different HMMs, one for each role.
- The structure of each HMM is set by hand, and follows the same basic form for each of the four different roles. Each state of an HMM is associated with a specific *label type*. The set of label types that is used, involve a *start* (S) label type that models the first token of the document, an *end* (E) type that models the last token, which is always the EOF (end of file) symbol, two *target* types (T_1 and T_2), which model the tokens that were hand tagged as one of the four target roles, two *prefix* (P_1 and P_2) and two *suffix* (S_1 and S_2) label types which model two tokens around the target tokens, and finally a *background* (B) type that models all the other tokens of the document which are of no particular interest. This set of label types is used to attribute a particular meaning to each state of the HMM, and it should not be confused with the token vocabulary *V* of the model. A typical HMM structure, using these label types is shown in Figure 1. The HMM of Figure 1 is not fully connected. This constraint on the allowable transitions encodes prior knowledge about the task, aiming to improve the extraction performance. For example, the self-transition in state "T2" indicates that a role instance, e.g. a *buyer* company, may consist of more than two tokens. Similarly, the transition from state T1 to state S1, indicates that a role instance may also consist of a single token.

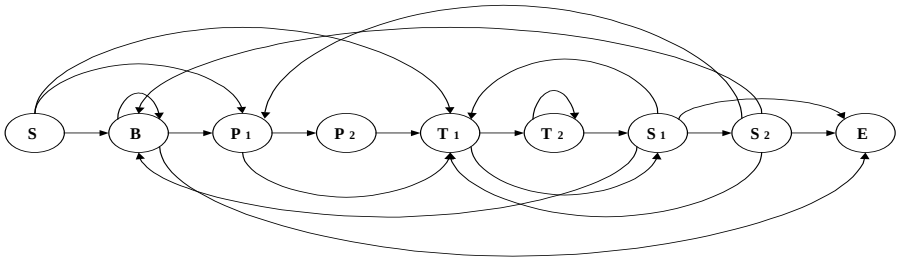

Fig. 1. An example of an HMM structure. Label types are associated to the states of the model (S: *start*, E: *end*, B: *background*, P1 *prefix1*, P2: *prefix2*, T1: *target1*, T2: *target2* S1: *suffix1*, S2: *suffix2*).

- A sequence of labels $L=\{l_1 l_2 \ldots l_T\}$ is associated with each training sequence $O=\{o_1 o_2 \ldots o_T\}$. L encodes the hand tagged information about the roles in a document, and l_i elements take values from the vocabulary of label types, as depicted in Figure 1. An example of a label sequence might be *L*={S B B…P1 P2

T1 T2 T2 S1 S2 B B…E}. When training an HMM for a specific role (e.g. *buyer company*), all tokens that are hand tagged with this role are associated with target tokens.

Since there is a one-to-one mapping between states and labels, the state sequence is no longer *hidden* and thus the Baum-Welch algorithm is not needed to train the HMMs. State transition and token emission probabilities can be acquired directly from the training data and their associated label sequences, by simply calculating ratios of counts (maximum likelihood estimation) as follows:

$$a_{ij} = \frac{c(i \to j)}{\sum_{s \in S} c(i \to s)} \text{ and } \quad b_j(k) = \frac{c(j \uparrow k)}{\sum_{v \in V} c(j \uparrow v)}. \tag{1}$$

Where $c(i \to j)$ counts the transitions from state *i* to state *j*, and $c(j \uparrow k)$ counts the occurrence frequency of token *k* in state *j*. Token emission probabilities often need to be *smoothed*, in order to avoid zero probabilities for vocabulary tokens not observed in the training data for a particular state. For that purpose we chose a widely used smoothing technique, described in [16]. State transition probabilities do not require smoothing, due to the small size and low connectivity of the model.

After the training phase, our four HMMs are evaluated using articles that have not been "seen" during the training process. Given a set of test sequences, each denoted as *O*, the objective is to find the most likely state sequence, i.e., the most likely label sequence *L*, and then extract the target tokens. The uncovering of the hidden label sequence corresponds to the second issue concerning HMMs, as described in subsection 2.1 and is achieved by the *Viterbi* algorithm. One issue that arises when following this modeling approach is how to deal with unknown tokens in the test collection, i.e., tokens that do not exist in the training vocabulary *V*. To deal with that problem we added a special token "unknown" to the vocabulary of the HMMs, during the training phase.

3. Experiments

3.1 Data preprocessing

For the purposes of our experiments we used a collection of 110 Greek financial articles describing company acquisition events. In these texts, the roles *buyer*, *acquired*, *rate* and *amount* were hand tagged. As mentioned above, *buyer* indicates the company that acts as a buyer, *acquired* indicates the company that is bought, the acquisition *rate* is the percentage of the company that is bought and finally the *amount* is the amount spent by the buyer. Each text describes a single company acquisition event. The text corpus was preprocessed using the *Ellogon* text

engineering platform [10] and the following linguistic tools: *tokenizer*, *part-of-speech-tagger* and *stemmer*.

The tokenizer identifies text tokens (i.e., words, symbols, etc.) and characterizes them according to a token-type tag set which encodes graphological information (e.g. the token comprises upper case greek characters). Part of this tag set is shown in Table 1(a). The part-of-speech (POS) tagger identifies the POS of each word token, according to a POS tag set. In addition to the part of speech, the tag set includes also morphological features, such as number, gender and case. Part of this tag set is shown in Table 1(b). The POS tagger that we used is a rule-based one, constructed with the use the transformation-based learning method [2]. The performance of the tagger on Greek financial texts is approximately 95% [9]. Finally, the stemmer converts word tokens to lowercase and unstressed, and removes the inflectional suffixes of Greek nouns and adjectives.

Table 1. (a) Part of the token-type subset used by the tokenizer. (b) Part of the part-of-speech tag set used by the POS tagger

GLW	Greek lowercase word
GUW	Greek uppercase word
GFW	Greek word first capital
EUW	English uppercase word
EFW	English word first capital
PERIOD	.
INT	Integer

(a)

NNF	Noun, feminine, singular number
DDT	Definite article
VBD	Verb past tense
WP	Relative pronoun
FW	Foreign word

(b)

The result of each linguistic processing step is a new collection of articles, where the linguistic information is represented as part of the text in various ways. Due to the sequential modeling nature of traditional HMMs, we represented the linguistic features of each token in sequence with the document text. For example, the result of the tokenizer is a new collection where an extra token is added just before each token, indicating the type of that token according to the tag set. Table 2 shows a sample sentence in the various data representations that we examined.

Table 2. Different representations for a sample sentence, incorporating linguistic information.

Collection A (Baseline)	Στην εξαγορά της εταιρείας ITC προχώρησε η Computer Logic.
Collection B (Type Token)	*GFW* Στην *GLW* εξαγορά *GLW* της *GLW* εταιρείας *EUW* ITC *GLW* προχώρησε *GLW* η *EUW* Computer *EUW* Logic *PERIOD* .
Collection C (POS Token)	*DDT* Στην *NNF* εξαγορά *DDT* της *NNF* εταιρείας *FW* ITC *VBD* προχώρησε *DDT* η *FW* Computer *FW* Logic.
Collection D (Type POS Token)	*GFW DDT* Στην *GLW NNF* εξαγορά *GLW DDT* της *NNF GLW* εταιρείας *EUW FW* ITC *GLW VBD* προχώρησε *GLW DDT* η *EUW* Computer *FW EUW FW* Logic *PERIOD* .

Collection E (Token_Type)	Στην_*GFW* εξαγορά_*GLW* της_*GLW* εταιρείας_*GLW* ITC_*EUW* προχώρησε_*GLW* η_*GLW* Computer_*EUW* Logic_*EUW* PERIOD .
Collection F (Token_POS)	Στην_*DDT* εξαγορά_*NNF* της_*DDT* εταιρείας_*NNF* ITC_*FW* προχώρησε_*VBD* η_*DDT* Computer_*FW* Logic_*FW* .
Collection G (Token_Type_POS)	Στην_*GFW_DDT* εξαγορά_*GLW_NNF* της_*GLW_DDT* εταιρείας_*GLW_NNF* ITC_*EUW_FW* προχώρησε_*GLW_VBD* η_*GLW_DDT* Computer_*EUW_FW* Logic_*EUW_FW* PERIOD .
Collection H (Type_POS)	*GFW_DDT GLW_NNF GLW_DDT GLW_NNF EUW_FW GLW_VBD GLW_DDT EUW_FW EUW_FW* PERIOD
Collection I (Stems)	στην εξαγορ της εταιρι itc προχωρησε η computer logic

3.2 Results

We conducted five groups of experiments. Each group uses collections from Table 2, which represent linguistic information in a similar manner. The first group contains experiments on the baseline collection A of Table 2 without any linguistic information. The second group contains experiments on the collections B, C and D, where the linguistic information (token type or POS or both) is represented as extra tokens added just before each token of the text. The third group contains experiments on the collections E, F and G, where the linguistic information is represented as tokens attached to each token of the text using the underscore character ("_"), as depicted in Table 2. The fourth group comprises the collection H where each token of the text is substituted by the corresponding type and POS, connected with each other using the underscore character (*Type_POS*). Finally, the fifth group contains the collection I, where each token from the baseline collection is substituted by the corresponding stem.

Each experiment on a collection, involves the training of four HMMs, one for each role of the domain. We experimented with various structures for the HMMs on each collection. The model structure, which achieved the best results for the majority of the collections, is shown in Figure 1. We conducted experiments using more than two prefix, suffix and target states, expecting that more complex HMM structures would capture the content of some collections where new tokens have been introduced, e.g. B, C and D, and thus achieve better results. However the results did not justify the additional complexity.

The evaluation of the HMMs was done using the 10-fold cross validation method. According to this evaluation method, the collection is split into ten equally sized subsets and the learning algorithm is run ten times. Each time, nine of the ten pieces are used for training and the tenth is kept as unseen data for the evaluation of the algorithm. Each of the ten pieces acts as the evaluation set in one of the ten runs and the final result is the average over ten runs. The extraction performance of the HMMs was evaluated using three measures per HMM (i.e., per role): *recall*, *precision* and *accuracy*. Recall measures the number of items of a certain role (e.g. buyer) correctly

identified, divided by the total number of items of this specific role in the data. Precision measures the number of items of a certain role correctly identified, divided by the total number of items that were assigned to that role by the HMM. Accuracy measures the number of *tokens* of a certain role correctly identified, divided by the total number of *tokens* assigned to that role [13]. In total 12 measures are used for the experiments: recall, precision and accuracy for each of the four roles (*buyer*, *acquired*, *rate*, *amount*) of the company acquisition domain.

The best results for each group of experiments together with the collections that achieved those results are presented in Tables 3 (a-e).

Table 3. Best performance of HMMs for each of the five groups of experiments

	Buyer	*Acquired*	*Rate*	*Amount*
Recall	0,294	0,238	0,856	0,517
Precision	0,567	0,531	0,791	0,397
Accuracy	0,721	0,617	0,806	0,607

(a) Performance on collection A (baseline collection)

	Buyer	*Acquired*	*Rate*	*Amount*
Best Collection	B	B	B	B
Recall	**0,637**	**0,571**	**0,967**	**0,592**
Precision	0,389	0,332	0,687	0,347
Accuracy	0,529	0,413	0,715	0,545

(b) Best performance on collections B, C, D

	Buyer	*Acquired*	*Rate*	*Amount*
Best Collection	G	G	G	G
Recall	**0,310**	**0,250**	0,838	**0,567**
Precision	**0,619**	**0,555**	**0,791**	**0,430**
Accuracy	**0,782**	**0,646**	**0,806**	**0,615**

(c) Best performance on collections E, F, G

	Buyer	*Acquired*	*Rate*	*Amount*
Recall	**0,697**	**0,683**	**0,915**	**0,842**
Precision	0,341	0,351	0,721	0,403
Accuracy	0,410	0,370	0,728	0,482

(d) Performance on collection H

	Buyer	*Acquired*	*Rate*	*Amount*
Recall	**0,309**	**0,286**	**0,856**	**0,567**
Precision	0,501	0,485	**0,796**	0,385
Accuracy	0,685	0,554	**0,814**	**0,631**

(e) Performance on collection I

Comparing the results in Table 3(b) to the baseline results in Table 3(a) we note a significant increase in recall, accompanied by a smaller decrease in both precision and

accuracy. This can be justified as follows: Capitalization of the first character of a token usually provides evidence of a name. By using the *Type Token* representation of collection B, our HMMs can learn rules of the form "when emitting one of *GUW*, *GFW*, etc., then with high probability the next token is a target token". Thus the number of items assigned to the *buyer* and *acquired* roles increases, causing the equivalent increase in recall, followed by a smaller decrease in the other two measures. On the other hand, the *rate* and *amount* roles mostly involve numerical entities. Thus the number of items assigned to those two roles also increases by the presence of a numeric token *Type*, e.g. *INT*, added just before a number. The learned rules in this case can be of the form "when emitting an integer or decimal number then with high probability the next token is a target token".

Comparing the results of Table 3(c) to the results of Table 3(a), we note an overall improvement for the *buyer*, *acquired* and *amount* roles, while the performance for the *rate* role remains almost unaffected. This means that the additional part-of-speech information included in this representation *(Token_Type_POS)* improves the performance of HMMs. The same is not true for collection D (*Type POS Token*), where the encoding of linguistic information as extra tokens causes a significant deterioration in precision and therefore the additional part-of-speech information is not beneficial.

When removing information about the token itself in collection H, the result is a significant increase in recall (comparing tables 3(d) and 3(c)), with a significant decrease in the other two measures. This is an indication of overgeneralization, which is expected due to the generality and simplicity of the linguistic information that is used, i.e., part-of-speech and token type.

The results in Table 3(e) show that stemming improves recall overall, while it hurts precision for the *buyer*, *acquired*, and *amount* roles. This means that the reduction of the vocabulary, with the use of stemming, causes a higher level of generalization, which increases recall and reduces precision. The performance for the *rate* role improves slightly in all three measures, which is justified by the emergence of clearer contextual patterns for this role, with the use of stemmed words.

Another clear conclusion from the experiments is that the performance for the *buyer* and *acquired* roles is worse than that for the *rate* role for all of the experiments. To a lesser extend the same is true for the *amount* role. There are two reasonable explanations for this. First, the *rate* and *amount* roles involve numerical entities, which are easier to detect in the text than detecting named entities, such as companies. This justifies the high recall for these roles. Second, it is more difficult to discriminate between roles for entities of the same type (e.g. *companies*). As a result many *buyer* companies in the collection were also detected by the HMMs as *acquired* and vice versa. On the other hand, *rate* and *amount* are very different from the other roles, and there aren't any other similar roles in the domain such as "*rate*_B" or "*amount*_B". This justifies the low precision for the *buyer* and *acquired* roles. To verify the second explanation we conducted another set of experiments where both *buyer* and *acquired* were tagged as one concept, i.e., "*buyer* OR *acquired*". The experiments were conducted using the collections with the best results in the previous experiments (Table 3). The new results are depicted in Table 4.

Table 4. Performance of the HMMs for the role "*buyer* OR *acquired*" on the collections with the best results from the five groups of experiments.

	Collection A (baseline)	Collection B (type token)	Collection G (token_type_pos)	Collection H (type_pos)	Collection I (stems)
Recall	0,432	0,738	0,426	0,842	0,416
Precision	0,611	0,400	0,620	0,560	0.560
Accuracy	0,758	0,579	0,767	0,648	0,741

As expected, in Table 4 a consistent improvement in all three measures (recall, precision and accuracy) is obtained over the results for the *buyer* and the *acquired* roles in Table 3. That improvement, however, is not as substantial as one would expect. This happens because there are also other company names in the collections that do not have a particular role in the acquisition event and the HMMs erroneously detect those entities as either *buyer* or *acquired*.

The ultimate question that remains unanswered is which representation leads to the best performance for HMMs? From the results of Table 3, we conclude that the best representation for the *buyer* and *acquired* roles is the one used in collection G (*Token_Type_POS*), which leads to a significant increase in all measures, in comparison to the baseline collection A. The representation of collection H (Type_POS) seems to achieve the best performance for the amount role. Finally, for the *rate* role the best representation seems to be the one used in collection I (*stems*). Note however that the *rate* is the role with the least deviation to the performance measures in all the experiments. This happens because the *rate* role involves exclusively numerical entities and the percent (%) symbol which are very little affected by the different representations used in the experiments. On the other hand, the amount role may further involve alphabetic characters (e.g. 40 εκ. δρχ.). Thus the performance for the amount role can be easier influenced by the various representations of Table 2.

4. Discussion and future work

In this paper we examined the effect of linguistic pre-processing of the training data to the performance of hidden Markov models in role identification. For the evaluation we used three measures (recall, precision and accuracy) and the 10-fold cross-validation method, in order to gain an unbiased estimate of the performance. The data that we used consisted of 110 Greek articles, announcing company acquisition events. These data were processed by simple and efficient linguistic analysis tools and were translated into training data for the HMMs using various representations, in which the linguistic information was represented as part of the text in a sequential form. The size of the HMMs that we used was small and their structure was simple, with the model parameters easily estimated from the training data, in a straightforward manner.

The experiments showed that using certain representations, simple linguistic analysis improves the extraction performance of HMMs on role identification. The overall performance was high for the two simpler roles (*rate* and *amount*), but it was

much lower for the other two roles (*buyer* and *acquired*). The improvement in performance gained by the use of linguistic information was clearer for the harder roles. The difficulty in identifying instances of the *buyer* and *acquired* roles stems mainly from the fact that they both correspond to the same type of entity (organization) and there is insufficient linguistic information for distinguishing between the two roles. Richer linguistic processing, involving syntactic analysis, could improve those results. This conclusion is also supported by the higher performance of an equivalent handcrafted system [5]. Indicative results of this system are shown in Table 5. The manual system performs badly for the *rate* and *amount* roles, due to the weak performance on the detection of numerical entities in text. However, it does much better in the other two roles, using much more extensive, albeit computationally expensive, linguistic analysis. Finally, our results are comparable to those reported in [3].

Table 5. Performance of a handcrafted system for the company acquisition domain

	Buyer	Acquired	Rate	Amount
Recall	75%	70%	49%	43%
Precision	72%	85%	72%	60%

The extraction performance of HMMs could be improved in several ways. Freitag & McCallum in [3], implemented a statistical technique called *shrinkage* that improves the token emission probabilities of an HMM in the presence of sparse training data. Furthermore, the learning of a probabilistic model such as an HMM, also involves the learning of the structure of the model. In this paper we assumed a fixed model structure, carefully designed for the particular dataset and domain that we used. However, certain structures may capture better the content of some documents straightforwardly affecting extraction performance. Machine learning techniques have been used for learning the structure of HMMs ([4], [13], [14]) from training examples.

Acknowledgments

This work was performed in the context of the CROSSMARC project, funded by the EC (contract IST-2000-25366), and has used data collected for the MITOS project, funded by the Greek Secretariat for Research and Technology (contract NEO EKBAN 2 1.3/102).

References

1. Bikel D.M., Miller S., Schwartz R., Weischedel R. (1997). *Nymble: a high performance learning name finder*. In Proceedings of ANLP-97, 194-201.
2. Brill E. (1995). *Transformation-Based Error Driven Learning and Natural Language Processing: A Case study in Part of Speech Tagging*, Computational Linguistics, vol. 21, n. 24.

3. Freitag D., McCallum A.K. (1999). *Information extraction using HMMs and shrinkage*. AAAI-99 Workshop on Machine Learning for Information Extraction, pp. 31-36. AAAI Technical Report WS-99-11.
4. Freitag D., McCallum A.K. (2000). *Information extraction with HMM structures Learned by Stochastic Optimization*, AAAI-2000, pp. 584-589.
5. Karkaletsis V., Farmakiotou D., Androutsopoulos I. Koutsias J., Paliouras G., Spyropoulos C.D. (2000). *Information Extraction from Greek Texts in the MITOS Information Management System.* Internal Technical Report, Institute of Informatics and Telecommunications, NCSR "Demokritos".
6. Kupiec, J. (1992). *Robust part-of-speech tagging using a hidden Markov model*. Computer Speech and Language, 6, 225-242.
7. Leek T.R. (1997). *Information extraction using hidden Markov models*, Master's thesis, UC San Diego.
8. MUC-6 (1995). *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufman, for Defense Advanced Research Projects Agency.
9. Petasis G., Paliouras G., Karkaletsis V., Spyropoulos C.D. and Androutsopoulos I. (1999). *Using Machine Learning Techniques for Part-of-Speech Tagging in the Greek Language*, Proceedings of the 7th Hellenic Conference on Informatics, Ioannina, Greece.
10. Petasis G., Karkaletsis V., Paliouras G., Androutsopoulos I., (2001). *Ellogon: A Text Engineering Platform.* Internal Technical Report, Institute of Informatics and Telecommunications, NCSR "Demokritos".
11. Rabiner, L., Juang B. (1986). *An introduction to hidden Markov models*. IEEE Acoustics, Speech & Signal Processing Magazine, 3 ,4-16.
12. Rabiner, L. (1989). *A tutorial on hidden Markov models and selected application in speech recognition*. Proceedings of the IEEE 1977 (2).
13. Seymore K., McCallum A., Rosenfeld R. (1999). *Learning hidden Markov model structure for information extraction.* AAAI-99 Workshop on Machine Learning for Information Extraction., pp. 37-42.
14. Stolcke A., Omohundro S. (1992). *Hidden Markov model induction by Bayesian model merging*. In Advances in Neural Information Processing Systems, volume 5. Morgan Kaufmann.
15. Yamron J., Carp I., Gillick L., Lowe S., Mulbregt P. (1998). *A hidden Markov model approach to text segmentation and event tracking*. In Proceedings of the IEEE ICASSP.
16. Witten, I. H., Bell T.C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory 37 (4).