International Journal on Artificial Intelligence Tools
© World Scientific Publishing Company

Combining Text Classifiers and Hidden Markov Models for Information Extraction

Flavia A. Barros
Center of Informatics, Federal University of Pernambuco
, Pobox 7851 - CEP 50732-970 - Recife (PE) - Brazil
fab@cin.ufpe.br

Eduardo F. A. Silva
Center of Informatics, Federal University of Pernambuco
, Pobox 7851 - CEP 50732-970 - Recife (PE) - Brazil
eduardo.amaral@gmail.com

Ricardo B. C. Prudêncio
Departament of Information Science, Federal University of Pernambuco
Av. dos Reitores, s/n - CEP 50670-901 - Recife (PE) - Brazil
prudencio.ricardo@gmail.com

In this paper, we propose a hybrid machine learning approach to Information Extraction by combining conventional text classification techniques and Hidden Markov Models (HMM). A text classifier generates a (locally optimal) initial output, which is refined by an HMM, providing a globally optimal classification. The proposed approach was evaluated in two case studies and the experiments revealed a consistent gain in performance through the use of the HMM. In the first case study, the implemented prototype was used to extract information from bibliographic references, reaching a precision rate of 87.48% in a test set with 3000 references. In the second case study, the prototype extracted information from author affiliations, reaching a precision rate of 90.27% in a test set with 300 affiliations.

Keywords: Information Extraction; Text Classifiers; HMM.

1. Introduction

With the growth of the Internet (in particular, the World Wide Web), we are challenged with a huge quantity and diversity of documents in textual format to be processed. This trend increased the difficulty to retrieve relevant data in an efficient way using traditional Information Retrieval methods [3]. When querying Web search engines or Digital Libraries, for example, the user has to go through the retrieved (relevant or not) documents one by one, looking for the desired data. *Information Extraction* (IE) appears in this scenario as a means to efficiently extract from the pre-selected documents only the information required by the user [2]. IE systems aim to identify parts of a document that correctly fill in a pre-defined template (output form).

This work focuses on IE from semi-structured text commonly available on the Web. Among the approaches to treating this kind of text (see section 2), we highlight the use of conventional Machine Learning (ML) algorithms for text classification [10], as this technique facilitates the systems customization to different domains. Here, the document is initially divided into fragments which will be later associated to slots in the output form by an ML classifier. Despite their advantages, these systems perform an independent classification for each fragment, disregarding the ordering of fragments in the input document [4].

Another successful classification technique used in the IE field are the Hidden Markov Models - HMM [16]. Different from the conventional classifiers, these models are able to take into account dependencies among the input fragments, thus favoring a globally optimal classification for the whole input sequence [4]. Nevertheless, the HMM can only treat one feature of each fragment, compromising local classification optimality [5].

With the aim of safeguarding the advantages of both techniques, we propose a hybrid ML approach, original in the IE field. In our work, a conventional ML algorithm generates an initial (locally optimal) classification of the input fragments that is refined by an HMM, providing a globally optimal classification for all text fragments.

In order to validate our approach, we implemented a prototype which was evaluated in two different domains: *bibliographic references*, aiming to extract information on author, title, publication date, etc; and *author affiliations*, aiming to extract information on department, institute, city, etc. The IE task is difficult in such domains, since their texts are semi-structured with high format variance [4].

The prototype was evaluated in these case studies through a number of experiments and revealed a consistent gain in performance with the use of the HMM. In the first case study, the gain due to the use of HMM ranged from 1.27 to 22.54 percentile points, depending on the classifier and on the set of features used in the initial classification phase. The best precision rate obtained was of 87.48% for a test set with 3000 references. In the second case study, the experiments confirmed the gain in performance with the HMM, which ranged from 3.53 to 14.12 percentile points. The best precision rate obtained in this case study was of 90.27% for a test set with 300 affiliations.

In a previous publication [17], we presented some preliminary experiments performed on the domain of bibliographic references. In the current paper, we present a more formal and detailed description of the proposed approach (Section 3), as well as the results of new experiments performed on the domains of bibliographic references and author affiliations.

The remainder of this paper is organized as follows: Section 2 presents techniques to IE; Section 3 details the proposed solution that combines conventional classification techniques and HMM; Section 4 discusses the experiments and results obtained in the case studies; Section 5 presents the conclusions and future work.

2. Information Extraction Techniques

"Information Extraction is concerned with extracting the relevant data from a collection of documents". An IE system aims to identify document fragments that correctly fill in slots (fields) in a given output template (form). The extracted data can be directly presented to the user or can be stored in appropriate databases.

The choice of the technique used to implement the IE system is strongly influenced by the kind of text in question. In the IE realm, text can be characterized as structured, semi-structured and non-structured (free text). Free text displays no format regularity, consisting of sentences in natural language. Contrarily, a structured text shows a rigid format (e.g., HTML pages automatically generated from databases). Finally, semi-structured text shows some degree of regularity, but may present incomplete information, variations in the order of the fields, no clear delimiters for the data to be extracted, and so on.

Natural Language Processing techniques are usually deployed to treat free text, as they are able to handle natural language irregularities [2]. On the other hand, Artificial Intelligence techniques, in particular Knowledge Engineering and Machine Learning (ML), have been largely used in IE from structured and semi-structured text.

Systems based on Knowledge Engineering usually reach high performance rates[14]. However, they are not easily customized to new domains, since they require the availability of domain experts and a large amount of manual work in rewriting rules. With the aim of minimizing these difficulties, a number of researchers use ML algorithms to automatically generate extraction rules from tagged corpora, thus favoring a quicker and more efficient customization of IE systems to new domains [18].

Among the ML systems used in the IE field, we initially cite those based on automata induction [11] and those based on pattern matching, which learn rules as regular expressions [18]. Systems based on these techniques represent rules using symbolic languages, that are, therefore, easier to interpret. However, they require regular patterns or clear text delimiters [18]. As such, they are less adequate for treating semi-structured texts which show a higher degree of variation in their structure (e.g., bibliographic references).

Different authors in the IE field have used conventional ML algorithms[a] as text classifiers for IE [105]. Initially, the input text is divided into fragments which will be later associated to slots in the output form. Next, an ML algorithm classifies the fragments based on their descriptive features (e.g., number of words, occurrence of an specific word, number or term, capitalized words, etc). Here, the class values

[a]Conventional ML algorithms include the Naive Bayes classifier [9] and the kNN algorithm [1], for instance.

correspond to the slots in the output form. The major drawback with these systems is that they perform a local and independent classification for each fragment, thus overlooking relevant structural information present in the document.

With the aim of minimizing the above-mentioned difficulty, a number of researchers have used HMMs to the IE task [10],[4]. These models are able to take into account dependencies among the input fragments, thus maximizing the probability of a globally optimal classification for the whole input sequence. Here, each slot (class) to be extracted is associated to a hidden state. Given a sequence of input fragments, the Viterbi algorithm [16] determines the most probable sequence of hidden states associated to the input sequence (i.e., which slot will be associated to each fragment). Nevertheless, despite their advantages, the HMMs can only consider one feature of each fragment (e.g., size, position or formatting) [5]. As said, this limitation compromises local classification optimality. More recently, Maximum Entropy Models [13] and Conditional Random Fields [12] have been applied to IE, extending the capabilities of the HMMs. However, the computational cost is a severe limitation to the use of these methods [6].

The following section presents a hybrid ML approach to the IE problem which combines conventional ML text classifiers and the HMMs, taking advantage of their positive aspects in order to increase the overall systems performance.

3. A Hybrid Approach for IE on Semi-structured Text

We propose here a hybrid approach for IE by combining conventional text classification techniques and HMMs to extract information from semi-structured text. The central idea is to perform an initial extraction based on a conventional text classifier and to refine it through the use of an HMM. By combining these techniques, we safeguard their advantages while overcoming their limitations. As mentioned above, conventional text classifiers offer a locally optimal classification for each input fragment, however disregarding the relationships among fragments. On the other hand, HMMs offer a globally optimal classification for all input fragments, but are not able to treat multiple features of fragments.

Figure 1 presents the extraction process performed by the proposed approach, illustrated in the domain of IE on bibliographic references. As it can be seen, the IE process consists of the following main steps:

(1) *Phase 1 - Extraction using a conventional text classifier.* This phase performs the initial extraction process, which is divided into three steps:
   (a) *Fragmentation of the input text.* The input text is broken into candidate fragments for filling in the output slots;
   (b) *Feature extraction.* A vector of features is created for describing each text fragment and is used in the classification of the fragment;
   (c) *Fragment classification.* A classifier decides (classifies) which slot of the output form will be filled in by each input fragment. This classifier is built via

a learning process that relates features of the text fragments and slots of the output form.

(2) *Phase 2 - Refinement of the results using an HMM.* The HMM receives as input the sequence of classes associated to the fragments in Phase 1 and provides a globally optimal classification of the input fragments.

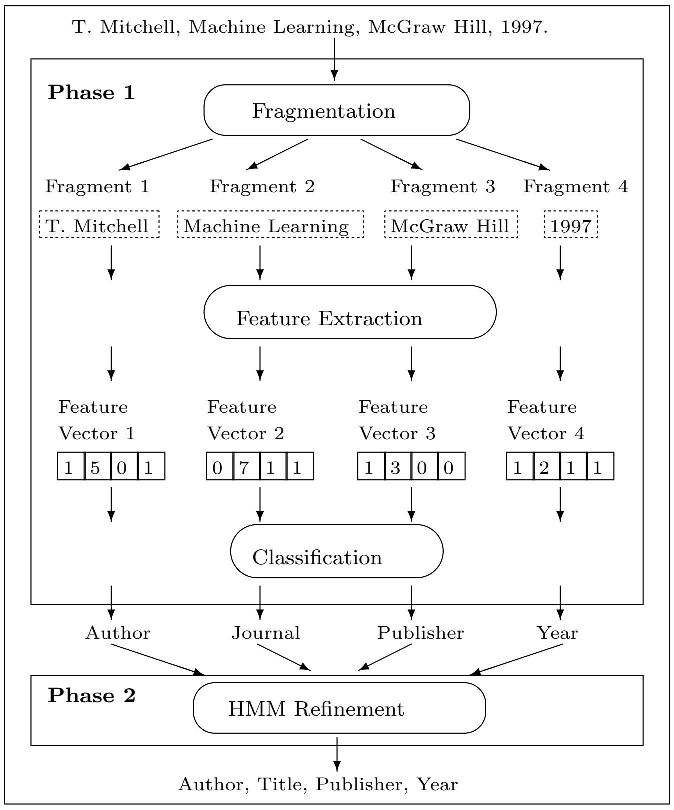

Fig. 1. Extraction process in the proposed approach

Our approach is strongly related to the Stacked Generalization technique [19], which consists of training a new classifier using as input the output provided by other classifiers, in a kind of meta-learning [8],[15]. However, our strategy is not to combine the output of different classifiers, but rather to use an HMM to refine the classification delivered by a single classifier for all input fragments.

The proposed combination is original in the IE field and has revealed very satisfactory experimental results in different case studies (see Section 4). The following subsections provide more details of the proposed extraction steps.

3.1. *Phase 1 - Extraction using a conventional text classifier*

This phase corresponds to the use of a conventional text classifier for IE (as mentioned in Section 2). It is divided into three steps, as follows.

3.1.1. *Fragmentation of the input text*

As said, the first step of this phase consists of breaking the input text into fragments that will be associated to slots in the output form. Let us now define this step in a more formal way.

Given an input text (or document) d, the fragmentation process generates an ordered sequence $(f_1, \ldots, f_M)$ composed by M text fragments. This segmentation is commonly performed by a set of heuristics that may consider text delimiters such as commas and other punctuation signs. Clearly, these heuristics are highly dependent on the application domain.

3.1.2. *Feature Extraction*

A vector of P features is computed for describing each text fragment, and it is used in the initial classification of the fragment. Formally, each fragment f_j is described by a vector of feature values $x_j = (x_j^1, \ldots, x_j^P)$, where $x_j^p = X_p(f_j)$ $(p = 1, \ldots, P)$ corresponds to the value of the descriptive feature X_p for the fragment f_j.

The set of descriptive features may be defined by a domain expert or automatically learned from a tagged corpus (see case studies in Section 4).

3.1.3. *Fragment Classification*

In this step, a classifier $\mathcal{L}$ associates each input fragment to one slot in the output form. Formally, for $(j = 1, \ldots, M)$, the classifier $\mathcal{L}$ receives the feature vector x_j describing the fragment f_j, and returns a class value $y_j \in \mathcal{C} = \{c_1, \ldots, c_K\}$, where each $c_k \in \mathcal{C}$ represents a different slot in the output form.

In what follows, we describe the training and use phases of the classifier.

(a) Classifier Training

In the proposed approach, the classifier $\mathcal{L}$ is built via a supervised learning process based on a training set $E^{[1]}$. Each training example in $E^{[1]}$ consists of a pair relating an input fragment to its correct slot in the output form.

$E^{[1]}$ is built based on a set of n texts (or documents) $D = \{d_1, \ldots, d_n\}$. Initially, each text $d_i \in D$ is automatically broken into M_i fragments $(f_{i,1}, \ldots, f_{i,M_i})$. Note that each document d_i in D may be broken into a different number of fragments. In the training examples, input fragments are actually represented by their corresponding feature vectors. Therefore, the following step consists of computing for each fragment $f_{i,j} \in d_i$ its feature vector $x_{i,j} = (x_{i,j}^1, \ldots, x_{i,j}^P)$, where $x_{i,j}^p = X_p(f_{i,j})(p = 1, \ldots, P)$ corresponds to the value of the feature X_p for the fragment $f_{i,j}$.

Finally, each fragment $f_{i,j} \in d_i$ is manually labeled with the class value $C_{i,j} \in \mathcal{C}$, which represents the fragment's correct slot in the output form. Formally, $E^{[1]} = \{(x_{i,j}, C_{i,j})\}$ $(i = 1, \ldots, n)$ $(j = 1, \ldots, M_i)$. The length of the training set $E^{[1]}$ corresponds to the total number of fragments obtained in the fragmentation of the n texts in D (i.e., $\sum_{i=1}^{n} M_i$).

We highlight that a large number of machine learning algorithms can potentially be used to learn the classifier in Phase 1. As it will be seen in Section 4, we opted to deploy only conventional ML algorithms, such as the kNN algorithm and the Naive Bayes classifier.

(b) Classifier Use Phase

After the training phase, the classifier $\mathcal{L}$ is used to predict a class value for each input fragment. Formally, given an input sequence of fragments $(f_1, \ldots, f_M)$, Phase 1 provides as output a sequence of class values $(y_1, \ldots, y_M)$, where $y_j = \mathcal{L}(x_j)$, $(j = 1, \ldots, M)$ is the class value predicted to the fragment f_j (here represented by its feature vector (x_j)).

3.2. *Phase 2 - Refinement of the results using an HMM*

This Phase is responsible for refining the initial classification yielded by Phase 1. This is performed by an HMM, which takes into account the order of the slots aiming to provide a globally optimal classification for the input fragments.

An HMM is a probabilistic finite automata that consists of: (1) a set of hidden states S; (2) a transition probability distribution in which $Pr[s'|s]$ is the probability of making a transition from the hidden state $s \in S$ to $s' \in S$; (3) a finite set of symbols T emitted by the hidden states; and (4) an emission probability distribution in which $Pr[t|s]$ is the probability of emitting the symbol $t \in T$ in state $s \in S$. Given an input sequence of symbols, the Viterbi algorithm [16] is used in the classification process to deliver the sequence of hidden states with the highest probability of emitting the input sequence of symbols.

In the proposed approach, given an input sequence of fragments, hidden states model their correct slots in the output form and the emitted symbols model the classes predicted by Phase 1. In this modelling, the Phase 2 receives as input the classes $(y_1, \ldots, y_M)$ predicted by Phase 1 for a sequence of fragments and returns the most likely sequence of correct slots.

Formally, the set of hidden states is defined here as $S = \{s_1, \ldots, s_K\}$ in such way that there is a one-to-one mapping between hidden states and class values. If the correct class of the j-th fragment is $c_k \in \mathcal{C}$, then the j-th state of the HMM is s_k. Similarly, the set of symbols is defined as $T = \{t_1, \ldots, t_K\}$, in such a way that, if the prediction of the Phase 1 for the j-th fragment is c_k then the j-th emitted symbol is t_k.

The transition probability $Pr[s_{k_1}|s_{k_2}]$ between the states s_{k_1} and s_{k_2} actually represents the probability that the correct class of a fragment is c_{k_1} given that the correct class of the previous fragment in the input text is c_{k_2}. Hence, the transition probability expresses the relationship between adjacent slots in the input texts. Formally, let d be an input text broken into M fragments $(f_1, \ldots, f_M)$, and let C_j $(j = 1, \ldots, M)$ be the correct class of the fragment f_j. The transition probability is defined as:

$$Pr[s_{k_1}|s_{k_2}] = Pr[C_{j+1} = c_{k_1}|C_j = c_{k_2}] \tag{1}$$

The emission probability $Pr[t_{k_1}|s_{k_2}]$, in turn, represents the probability that the classifier of Phase 1 predicts the class value c_{k_1}, given that the correct class of the fragment is c_{k_2}. The emission distribution tries to capture regularities in the errors occurring in the classification yielded by Phase 1. Formally, let y_j be the prediction provided by Phase 1 for a fragment f_j and let C_j be the correct slot of this fragment. The emission probability is defined as:

$$Pr[t_{k_1}|s_{k_2}] = Pr[y_j = c_{k_1}|C_j = c_{k_2}] \tag{2}$$

(a) HMM Training

The training of the HMM consists of estimating the transition and emission probabilities. This is performed by a supervised learning process using a training set $E^{[2]}$ which is built based on the same set $D = \{d_1, \ldots, d_n\}$ of n texts used in Phase 1. Each training example in $E^{[2]}$ is related to one single text $d_i \in D$, and consists of a sequence of pairs containing the class predicted to each text fragment $f_{i,j}$ in Phase 1 and the class to which the fragment actually belongs.

Let $(f_{i,1}, \ldots, f_{i,M_i})$ be the sequence of M_i fragments obtained from the text $d_i \in D$, and let $(y_{i,1}, \ldots, y_{i,M_i})$ be the sequence of corresponding classes predicted in Phase 1. Then, $E^{[2]} = \{((y_{i,1}, C_{i,1}), \ldots, (y_{i,M_i}, C_{i,M_i}))\}$ $(i = 1, \ldots, n)$, where $C_{i,j}$ corresponds to the correct class of the fragment $f_{i,j}$. The total number of training sequences corresponds to the number of texts in set D.

The transition probability for a given pair (s_{k_1}, s_{k_2}) of hidden states is estimated by computing the ratio of: (1) the number of transitions from s_{k_2} to s_{k_1} observed in $E^{[2]}$ (i.e., the number of adjacent fragments $f_{i,j+1}$ and $f_{i,j}$ that respectively belong to classes c_{k_1} and c_{k_2}) to (2) the total number of transitions from s_{k_2} (i.e., the total number of fragments $f_{i,j}$ that belong to class c_{k_2}). This ratio can be defined as:

$$Pr[s_{k_1}|s_{k_2}] = \frac{\#[(C_{i,j+1} = c_{k_1})\ AND\ (C_{i,j} = c_{k_2})]}{\#[(C_{i,j} = c_{k_2})]} \tag{3}$$

The emission probability for a given symbol t_{k_1} and hidden state s_{k_2} is estimated by the ratio of: (1) the number of times that the symbol t_{k_1} was emitted by s_{k_2} (i.e., the number of fragments $f_{i,j}$ classified by Phase 1 as c_{k_1}, but that actually belong to class c_{k_2}) to (2) the total number of emissions from s_{k_2} (i.e., the total number of fragments $f_{i,j}$ that belong to c_{k_2}). This ratio can be defined as:

$$Pr[t_{k_1}|s_{k_2}] = \frac{\#[(y_{i,j} = c_{k_1})\ AND\ (C_{i,j} = c_{k_2})]}{\#[(C_{i,j} = c_{k_2})]} \tag{4}$$

(b) HMM Use Phase

As said, given the transition and emission probabilities, the HMM can be used to associate sequences of hidden states to input sequences of symbols. In Phase 2, given the sequence of symbols corresponding to the class values $(y_1, \ldots, y_M)$ provided by Phase 1, the Viterbi algorithm delivers a sequence of hidden states that is mapped onto the final output of Phase 2.

Formally, Phase 2 consists of a classifier $\mathcal{HMM}$, and the refined output $(\widetilde{y}_1, \ldots, \widetilde{y}_M)$ for the whole sequence of text fragments is defined as:

$$(\widetilde{y}_1, \ldots, \widetilde{y}_M) = \mathcal{HMM}(y_1, \ldots, y_M) \tag{5}$$

In the following section, we present the case studies that evaluated the proposed solution. As it will be seen, the performed experiments revealed a consistent gain in performance when outputs of the $\mathcal{HMM}$ classifier are compared to the outputs provided solely by Phase 1.

4. Case Studies

In this section, we present two case studies that evaluate the viability of the proposed solution for the IE task. Section 4.1 presents the IE on bibliographic references and section 4.2, in turn, presents the IE on author affiliations.

4.1. *Case Study 1: IE on Bibliographic References*

The first case study tackled the problem of IE from bibliographic references. The motivation for choosing this domain was the automatic creation of research publication databases, very useful to the scientific and academic communities. Information that can be extracted from a bibliographic reference includes author(s), title, date of publication, among others.

Bibliographic references are semi-structured texts with a high degree of variation in their structure. This turns the IE in this domain into a difficult task [4]. Examples of such variations are: (1) the fields can appear in different orders (e.g., author can be the 1st or the 2nd field); (2) absent fields (e.g., the pages of a paper are sometimes omitted); (3) telegraphic style (e.g., pages can be represented by “pp”); (4) absence of precise delimiters (some delimiters, such as “,” and “.”, may appear inside a field to be extracted).

We present bellow the implementation details regarding this case study, followed by the performed experiments and results.

4.1.1. *Implementation Details*

In what follows, we present details on how the two phases of the proposed approach were designed and implemented in this case study.

(1) Phase 1 - Classification Using Conventional Text Classifiers

(a) *Fragmentation of the input text*: Here, we deployed heuristics based on commas and punctuation.

(b) *Feature extraction*: we used here three distinct sets of features for describing the text fragments. Two sets define features specific to the domain of references (the set used in the ProdExt system [14] and the set defined by [5]). These two sets were defined through a knowledge engineering process and contain characteristics such as the occurrence of specific terms (e.g., "journal" and "vol"), publisher names, dates indicated as years, etc. The third set contains words directly obtained from a corpus of bibliographic references and selected using the Information Gain method [20].

(c) *Fragment classification*: we defined 14 different slots for the domain of references: author, title, affiliation, journal, vehicle, month, year, editor, place, publisher, volume, number, pages, and others. In this step, we used three classifiers, each representing a family of ML algorithms: the Naive Bayes [9], the PART (Rules) algorithm [7] and the k-NN [1]. These algorithms were implemented using the WEKA environment[b].

(2) Phase 2 - Refinement of the Results Using an HMM

As previously mentioned, Phase 1 classifies the text fragments independently from each other. However, the information to be extracted from a reference follows an ordering that, although not rigid, may help the extraction process to provide a global optimal classification of the input fragments. To take advantage of this structural information, the output delivered by the classifier in Phase 1 is refined by an HMM in order to correct errors in the initial classification.

Figure 2 presents a simplified HMM containing 3 symbols (represented by rectangles), and 3 hidden states (represented by circles), each one identified by the name of the slot to which it is associated. In this case study, all hidden states were connected to each other, since the correct classification of each input fragment may be related to the classification of the other fragments.

Figure 3 presents training examples of the HMM. Each example consists of a sequence of pairs containing the class associated to a fragment in Phase 1 and the class to which it actually belongs. In Example 1 of Figure 3, the second fragment of a given reference was classified in Phase 1 as Journal, but it actually belongs to the Title class. In Example 2, all fragments of a reference were correctly classified; and in Example 3, the third fragment was classified as Author rather than Editor.

4.1.2. *Experiments and Results*

The implemented prototype was trained and tested using a corpus from the Bibliography on computational linguistics, systemic and functional linguistics, artificial

[b]Weka 3: Data Mining Software in Java - http://www.cs.waikato.ac.nz/~ml/weka/

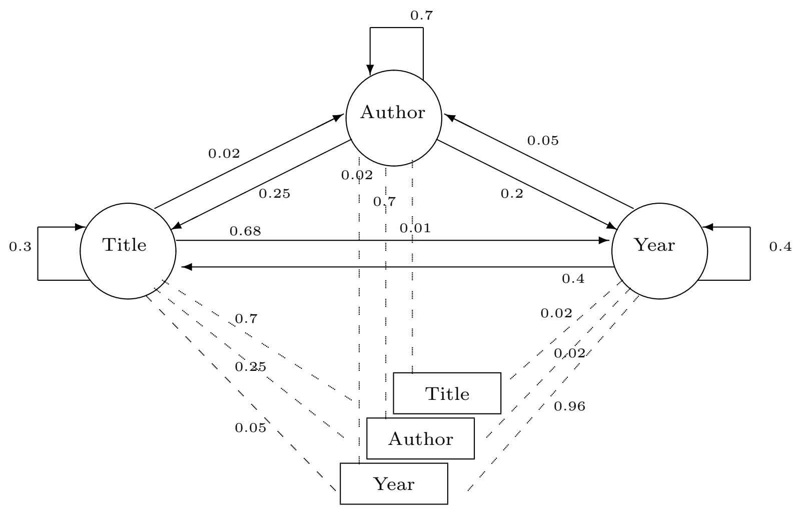

Fig. 2. Example of HMM in the refinement phase

1: ((Author, Author), (Journal, Title), (Vehicle, Vehicle), (Year, Year))

2: ((Author, Author), (Title, Title), (Year, Year), (Place, Place))

3: ((Author, Author), (Title, Title), (Author, Editor), (Year, Year))

Fig. 3. Examples of sequences in set $E^{[2]}$ used for the HMM training.

intelligence and general linguistics collection, which contains 6 thousand bibliographic references with tags that indicate the class of each text fragment[c]. The average number of fields per reference was 6.22, showing that the 14 slots on the output form do not always appear in the references (the most frequent are Author, Title and Year). The collection of references was divided equally into two sets of 3000 references, one for training and the other for testing the system's performance.

The experiments evaluated the performance of our IE system with HMM refinement compared to the system without the HMM. The following aspects were considered: the feature set; and the classifier used in Phase 1. As seen in Section 4.1.1, three classifiers were used here: the Naive Bayes, the PART (Rules) and the kNN.

In these experiments, we tested 6 combinations of the feature sets cited in Section 4.1.1: (1) Manual1 (20 features used in the ProdExt system [14]); (2) Manual2 (9 features defined by [5]); (3) Automatic (100 terms selected from the training cor-

[c]Available on-line at http://liinwww.ira.uka.de/bibliography/Ai/bateman.html.

pus); (4) Manual1+ Manual2 (27 features); (5) Automatic + Manual2 (total of 109 features); and (6) Automatic + Manual1 + Manual2 (127 features).

Each combination represents a different level of expertise that is required to define the features. The combination Manual1+Manual2, for instance, represents the maximum level of expertise, as it combines two sets defined by a human expert. The Automatic+Manual2 set, on its turn, represents an intermediate level of expertise effort. We point out that this combination (i.e., features defined by an expert and features automatically selected from a training corpus) is original for the IE task on bibliographic references.

The system's performance was evaluated with the use and without the use of the HMM for each combination of feature set versus classifier. The evaluation measure used was precision, defined as the number of correctly extracted slots divided by the total number of slots present in the references.

Table 1 shows the average precision per fragment obtained for each combination of feature set versus classifier. By comparing the precision obtained with and without the HMM, we verified a gain in performance with the use of HMM in all combinations. The gain varied from 1.27 to 22.54 percentage points. The best result was a precision of 87.48%, obtained using the set Automatic+Manual2, the classifier PART and the refinement with the HMM.

Feature Set	Classifier	Precision without HMM	Precision with HMM	Gain
Manual1	PART	72.17%	76.40%	4.22%
Manual1	Bayes	66.70%	74.72%	8.01%
Manual1	kNN	71.96%	76.28%	4.32%
Manual2	PART	73.48%	77.29%	3.80%
Manual2	Bayes	69.03%	77.27%	8.23%
Manual2	kNN	76.17%	81.16%	4.99%
Automatic	PART	49.91%	72.45%	22.54%
Automatic	Bayes	50.11%	68.25%	18.14%
Automatic	kNN	51.47%	73.57%	22.10%
Manual1+Manual2	PART	81.99%	86.00%	4.00%
Manual1+Manual2	Bayes	71.89%	81.43%	9.54%
Manual1+Manual2	kNN	81.40%	83.21%	1.81%
Automatic+Manual2	PART	83.74%	87.48%	3.75%
Automatic+Manual2	Bayes	74.78%	83.46%	8.69%
Automatic+Manual2	kNN	83.23%	84.85%	1.62%
Automatic+Manual1+Manual2	PART	84.82%	87.36%	2.54%
Automatic+Manual1+Manual2	Bayes	75.29%	84.20%	8.90%
Automatic+Manual1+Manual2	kNN	83.89%	85.17%	1.27%

The system's performance significantly varied depending on the classifier used in Phase 1. For all the feature sets used, we observed a lower average performance using the Naive Bayes classifier, especially without the use of HMM (see Table 2). However, we observed that the use of the HMM improved the low performance of this classifier, delivering results closer to those obtained by the other classifiers. Hence, we conclude that the variability of the system performance, considering the classifier used in Phase 1, is lower when the HMM is used in the refinement phase.

The set of features used in Phase 1 also strongly influenced system performance. The Automatic set issued the worst average precision rate (see Table 3). However, system performance using this set was clearly improved by the use of the HMM, coming closer to the results issued by the other sets of features. This is considered as evidence that the HMM is able to compensate the use of less expressive feature sets, such as the automatically created sets, thus facilitating the customization of the system to different IE domains.

Classifier	Average Precision without HMM	Average Precision with HMM
PART	74.35%	81.16%
Bayes	67.97%	78.22%
KNN	74.69%	80.71%

Classifier	Average Precision without HMM	Average Precision with HMM
Manual1	70.27%	75.80%
Manual2	72.89%	78.57%
Automatic	50.49%	71.42%
Manual1+Manual2	78.42%	83.54%
Automatic+Manual2	80.58%	85.26%
Automatic+Manual1+Manual2	81.33%	85.57%

4.2. *Case Study 2: IE on Author Affiliations*

This section presents a second case study that further evaluated the proposed approach. This case study focused on the author affiliation domain, aiming to extract information such as author, name of the department, zipcodes, street, country, city, among others. Here, the IE system helps the process of converting scientific documents written by different authors to an uniform structured format [5], which facilitates the retrieval of these documents.

Similarly to bibliographic references, author affiliations present variations in their structure that hardens the extraction task. Examples of these variations are: (1) fields in different orders (e.g., name of the department can appear after the name of the institute, and vice-versa); (2) absent fields (e.g., authors in USA commonly omit the country name); (3) telegraphic style (e.g., “Depart”, “Univ”).

Following, we present implementation details regarding this case study, and the experiments and results.

4.2.1. *Implementation Details*

We present here the implementation details regarding this case study, following the same structure of section 4.1.

(1) *Phase 1 - Classification Using Conventional Text Classifiers*

(a) *Fragmentation of the input text*: according to [5], author affiliations have the property that items to be extracted are commonly separated by commas, which allows a simple fragmentation process. Hence, we deployed here a single heuristic based on the occurrence of commas, spaces and punctuation.

(b) *Feature extraction*: in this case study, we used two sets of features. The first set was defined by [5] and contains domain characteristics such as occurrence of specific terms (e.g., "department" and "university"), names in a list of cities and countries, regular expressions matching Zip code, among others. The second set contains the words selected using the Information Gain method, as performed in the first case study.

(c) *Fragment classification*: we defined 7 slots for the domain of affiliations: street, pobox, city, zip, country, department and institute. As in the first case study, we used here the WEKA implementation of the Naive Bayes, the PART (Rules) and the kNN algorithms.

(2) *Phase 2 - Refinement of the Results Using an HMM*

As in the bibliographic reference domain, the information to be extracted from paper affiliations presents an ordering that may help the extraction process. For example, the name of the department and university commonly correspond to adjacent fragments in the affiliation. Hence, the use of an HMM in the refinement phase may provide a gain in the overall extraction performance. The structure of the HMM in this case study was also defined with all hidden states connected to each other.

4.2.2. *Experiments and Results*

The prototype was trained and tested in the second case study using a corpus of 600 affiliations from computer and information science papers collected from the CiteSeer metadata[d]. The average number of fields per affiliation was 4.59, and the most frequent fields were Institute, City, Zip and Department. The collection was equally divided into two sets of 300 affiliations, respectively for training and testing of the system's performance.

As in the first case study, the experiments evaluated the performance of the IE system with and without the HMM refinement, considering the feature set and the

[d]http://citeseer.ist.psu.edu/oai.html

classifier used in Phase 1 (Naive Bayes, PART (Rules) and kNN).

In the experiments, we tested 3 combinations of the feature sets cited in Section 4.2.1: (1) Manual (10 features defined by [5]); (2) Automatic (100 terms selected from the training corpus); and (3) Automatic + Manual (total of 110 features). As said, each combination represents a different level of expertise that is required to define the features.

Table 4 shows the average precision obtained for each combination of feature set and classifier. As in the first case study, we observed a performance gain in all combinations when the HMM was used (the gain varied from 3.53 to 14.12 percentage points). Similarly to the first case study, the best result (90.27%) was obtained using a combined feature set, the classifier PART and the refinement with the HMM.

Tables 5 and 6 show that the precision performance varied depending on both the classifier and the feature set used in Phase 1. Nevertheless, we observed that the variation of the performance is lower when the HMM is used. As in the first case study, the performance of the IE system was less dependent on the selection of a more adequate learning technique and a more expressive set of features.

Feature Set	Classifier	Precision without HMM	Precision with HMM	Gain
Manual	PART	68.87%	82.99%	14.12%
Manual	Bayes	67.65%	80.11%	12.46%
Manual	kNN	69.23%	80.54%	11.31%
Automatic	PART	57.27%	64.04%	6.77%
Automatic	Bayes	67.36%	74.13%	6.77%
Automatic	kNN	69.88%	74.78%	4.90%
Automatic + Manual	PART	86.74%	90.27%	3.53%
Automatic + Manual	Bayes	84.94%	90.12%	5.18%
Automatic + Manual	kNN	85.37%	89.62%	4.25%

Classifier	Precision without HMM	Precision with HMM
PART	70.96%	79.10%
Bayes	73.32%	81.45%
KNN	74.83%	81.65%

Classifier	Precision without HMM	Precision with HMM
Manual	68.58%	81.21%
Automatic	64.84%	70.98%
Automatic+Manual	85.68%	90.00%

5. Conclusion

We propose here a hybrid machine learning approach to the IE problem based on the combination of traditional text classifiers and HMMs. The main contribution of this work is to have joined two techniques not yet combined in an IE system. Here, the HMM is used to refine the initial classification issued by the text classifier. In the experiments performed on two different case studies, we observed that the use of an HMM compensated the low performance of less adequate classifiers and feature sets chosen to implement the text classifier.

As future work, we intend to improve the results obtained in the case studies by automatically defining the HMM structure, and evaluating new classifiers and features sets in the initial text classification. We also intend to evaluate the impact of the text fragmentation step in the IE process, and to investigate the use of machine learning to induce fragmentation rules.

References

1. D. Aha and D. Kibler, Instance-based learning algorithms, *Machine Learning*, Vol.6(3), 1991, pp. 37–66.
2. D. E. Appelt and D. Israel, Introduction to information extraction technology, *International Joint Conference on Artificial Intelligence (IJCAI-99) Tutorial*, Stockholm, Sweden, 1999.
3. R. Baeza-Yates and B. Ribeiro Neto, *Modern Information Retrieval*, Addison-Wesley, New York, EUA, 1999.
4. V. R. Borkar, K. Deshmukh and S. Sarawagi, Automatic segmentation of text into structured records, *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, 2001, pp. 175–186.
5. R. R. Bouckaert, Low level information extraction: a bayesian network based approach, *TextML*, 2002.
6. T. G. Dietterich, Machine learning for sequential data: a review, *Lecture Notes in Computer Science*, Vol. 2396, 2002, pp. 15–30.
7. E. Frank and I. H. Witten, Generating accurate rule sets without global optimization, *Proceedings of the 15th International Conference on Machine Learning*, 1998, pp. 144–151.
8. C. Giraud-Carrier, R. Vilalta and P. Brazdil, Introduction to the special issue on meta-learning, *Machine Learning*, Vol. 54(3), 2004, pp. 187–193.
9. G. H. John and P. Langley, Estimating continuous distributions in bayesian classifiers, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Mateo, 1995, pp. 338–345.
10. N. Kushmerick, E. Johnston and S. McGuinness, Information extraction by text classification, *IJCAI-01 Workshop on Adaptive Text Extraction and Mining*, Seattle, WA, 2001.
11. R. Kosala, J. Bussche, M. Bruynooghe and H. Blockeel, Information Extraction in Structured Documents Using Tree Automata Induction, *Lecture Notes on Computer Science*, Vol. 2431, 2002, pp. 299–310.
12. J. Laferty, A. McCallum and F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *International Conference on Machine Learning*, San Francisco, CA, 2001.
13. A. McCallum, D. Freitag and F. Pereira, Maximum entropy Markov models for infor-

mation extraction and segmentation, *International Conference on Machine Learning*, pp. 591-598, Morgan Kaufmann, 2000.

14. C. Nunes and F. A. Barros, ProdExt: a knowledge-based wrapper for extraction of technical and scientific production in web pages, *Proceedings of the International Joint Conference IBERAMIA-SBIA 2000*, 2000, pp. 106-115.
15. R. B. C. Prudêncio and T. B. Ludermir, Meta-learning approaches for selecting time series models, *Neurocomputing Journal*, Vol. 61(C), 2004, pp. 121–137.
16. L. R. Rabiner and B. H. Juang, An introduction to Hidden Markov Models, *IEEE ASSP Magazine*, Vol. 3(1), 1986, pp. 4–16.
17. E. Silva, F. Barros and R. Prudêncio, A Hybrid Machine Learning Approach for Information Extraction, *Proceedings of the 6th International Conference on Hybrid Intelligent Systems*, Auckland, New Zealand, 2006 (to appear).
18. S. Soderland, Learning information extraction rules for semi-structured and free text, *Machine Learning*, Vol. 34(1-3), 1999, pp. 233–272.
19. D. H. Wolpert, Stacked generalization, *Neural Networks*, Vol. 5, 1992, pp. 241–259.
20. Y. Yang and J. O. Pedersen, A comparative study on feature selection methods in text categorization, *Procceedings of the 14th International Conference on Machine Learning*, 1997, pp. 412–420.