Hidden Markov Models and Text Classifiers for Information Extraction on Semi-Structured Texts

Flavia A. Barros, Eduardo F. A. Silva, Ricardo B. C. Prudêncio
Valmir M. Filho and André C. A. Nascimento
Center of Informatics
Federal University of Pernambuco
Pobox 7851 - CEP 50732-970 - Recife (PE) - Brazil
{fab, efas, rbcp, vmf2, acan}@cin.ufpe.br

Abstract

Information Extraction (IE) aims to extract from textual documents only the fragments which correspond to data fields required by the user. In this paper, we present new experiments evaluating a hybrid machine learning approach for IE that combines text classifiers and Hidden Markov Models (HMM). In this approach, a text classifier technique generates an initial output, which is refined by an HMM, taking into account dependences in the order of the data to be extracted. The proposal was evaluated to extract information from bibliographic references. Experiments performed on a corpus of 6000 references have shown an improvement in performance compared to benchmarking IE approaches adopted in previous work.

1 Introduction

The huge amount of textual documents available in digital repositories (such as, the World Wide Web) has increased the difficulty to retrieve relevant data by using traditional Information Retrieval (IR) methods [2]. When querying Web search engines (e.g., Google), for instance, the user has to browse the retrieved (relevant or not) documents one by one, looking for the desired data. In this context, *Information Extraction* (IE) systems arise as a means to facilitate the information access, by extracting from the documents only the parts that correctly fill in a set of pre-defined output slots (data fields) [27]. The extracted data can be directly presented to the user or can be stored in appropriate databases.

Among the approaches for IE, we highlight the use of Machine Learning (ML) algorithms as text classifiers [15]. In this approach, the input document is initially divided into fragments which will be later associated to the output slots by a text classifier. The classification is performed based on descriptive features of the fragment (e.g. its length, presence of terms, etc). Despite their advantages, these systems classify each input fragment independently on the other fragments. As such, they miss important information about the document's structure [3].

In order to minimize the above difficulty, a hybrid IE approach was proposed by combining traditional ML text classifiers and Hidden Markov Models (HMMs) [21] in a two-phase architecture. In this proposal, given an input document, a ML text classifier generates an initial classification of the document's fragments. Following, the HMM receives the whole sequence of outputs provided by the text classifier in the first phase, and returns a refined classification by taking into account dependencies among the input fragments.

In [23], the authors provided the initial experiments which evaluated the viability of the proposal. In the current work, we present more discussion and new experiments with the proposed approach, including the use of Support Vector Machines in the first phase of classification and a comparison to a benchmarking IE approach successfully used in previous work [4].

The proposed approach was evaluated in a case study which corresponds to the task of IE on bibliographic references, aiming to extract information such as author, title, year, etc. A bibliographic reference is seen as a semi-structured text with a high variance in its structure [3]. The prototype was evaluated through a number of experiments with different configurations (e.g., with different text classifiers) and revealed a consistent gain in performance compared to other IE approaches in a corpus of 6000 references. We also observed that the proposed system revealed to be less sensitive to a good choice of the set of features and the algorithm used in the initial classification of the text fragments.

Section 2 presents techniques to IE. Section 3 brings de-

tails of the proposed solution. Section 4 describes the case study as well as the experiments and obtained results. Finally, section 5 presents some conclusions.

2 Information Extraction

Information Extraction (IE) is concerned with extracting relevant data from a collection of documents [27]. An IE system identifies document fragments that correctly fill in slots in a given output form. The extracted data can be directly presented to the user or stored in a database to be latter accessed in structured interfaces.

Machine Learning (ML) techniques have been largely used for IE in order to automatically generate extraction rules from tagged corpora. Among the ML systems for IE, we cite those based on the learning of finite automata and regular expressions [14, 18, 8]. Systems based on these techniques represent rules using symbolic languages that are easier to interpret. However, they require regular patterns or clear text delimiters and hence are less adequate for texts which show a higher degree of variation in structure [26].

An alternative approach for IE is the use of conventional ML algorithms[1] as text classifiers [31, 6, 15, 24, 16]. Initially, the input text is divided into fragments which will be later associated to the output slots. Next, an ML algorithm classifies each fragment based on its descriptive features (e.g., number of words, occurrence of numbers, etc). Here, the class values 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 drawbacks, a number of researchers have used sequential learning algorithms [7] to the IE task. In this context, we mention the *Sliding Window* method, which has been successfully used, for instance, in [4]. This approach is similar to a conventional text classifier, however the classification of each fragment is not solely based on its own features, but it also considers the features of adjacent fragments. Hence, the classifier actually receives as input a *window* defined on the sequence of fragments to be classified. The *width* of the input window defines the number of adjacent fragments that must be considered by the classifier. For instance, considering an input window of width 3, a fragment classification must be performed based on its own features plus the features of its just precedent and following fragments.

Another sequential learning algorithm that has been successfully applied to some IE tasks is the Hidden Markov Models (HMMs) [15, 3, 25]. 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 [21] 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 deal with fragments which are single tokens of the input text and typically consider one feature of each token [4]. Hence, the HMMs are not straightforwardly suitable for IE tasks in which many properties can be assigned to the fragments. As said, this limitation compromises local classification optimality.

3 Combining HMMs and Text Classifiers

In this work, we propose the combination of Hidden Markov Models and text classifiers for IE. In this proposal, an initial extraction performed by a conventional text classifier is refined through the use of an HMM. As mentioned, 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. The proposed approach is more suitable for IE on semi-structured texts showing some degree of regularity in the sequence of the slots to be extracted, but which may present some difficulties, such as incomplete fields and variations in the order of the fields.

Our approach is strongly related to the Stacked Generalization technique [29], which consists of training a new classifier using as input the output provided by other classifiers, in a kind of meta-learning [12]. 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.

We can also mention as related work, other hybrid ML approaches for IE presented in the literature. In [15], the authors combined a probabilistic text classifier and HMMs for IE. Different from [15], however, our proposal is not restricted to the use of probabilistic text classifiers. In [10, 19, 22], for instance, the outputs of different text classifiers for IE were combined by using voting schemas. In [11], a boosting mechanism was applied to combine different IE systems. In [22], the authors developed a stacked generalizer for IE, which is similar to our proposal, however only non-sequential algorithms (such as conventional decision trees) were deployed in the second step of extraction. In [8], the authors developed a hybrid ML approach for IE in which a knowledge-based system and a grammar inference algorithm were combined in order to derive extraction

[1] Conventional ML algorithms may be for instance the Naive Bayes classifier and the kNN algorithm.

rules.

Figure 1 presents the proposed approach, illustrated in the domain of 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.* The initial extraction process is divided into:

   (a) *Fragmentation of the input text.* The input text must be divided into candidate fragments for filling in the output slots. This segmentation is commonly performed by a set of heuristics that may consider text delimiters.

   (b) *Feature extraction*. A vector of features is created for describing each fragment and is used in the classification of the fragment.

   (c) *Fragment classification.* A classifier decides which output slot will be filled in by each input fragment. Here, we build conventional ML algorithms by using a corpus of tagged fragments as training set.

2. *Phase 2 - Refinement of the results using an HMM*. The HMM refines the initial extraction, providing a globally optimal classification for the whole sequence of 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 $s \in S$. The Viterbi algorithm is used in the classification process, delivering a sequence of hidden states with the highest probability of generating each input sequence of symbols. The HMM may induces the probability distributions $Pr[s'/s]$ and $Pr[t/s]$ by the use of a training set that associates hidden states and emitted symbols.

Here, each hidden state represents an output slot, and the emitted symbols represent the classes predicted by Phase 1. Formally, let $\mathcal{C} = \{c_1, \ldots, c_K\}$, where each $c_k \in \mathcal{C}$ represents a different slot in the output form. 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.

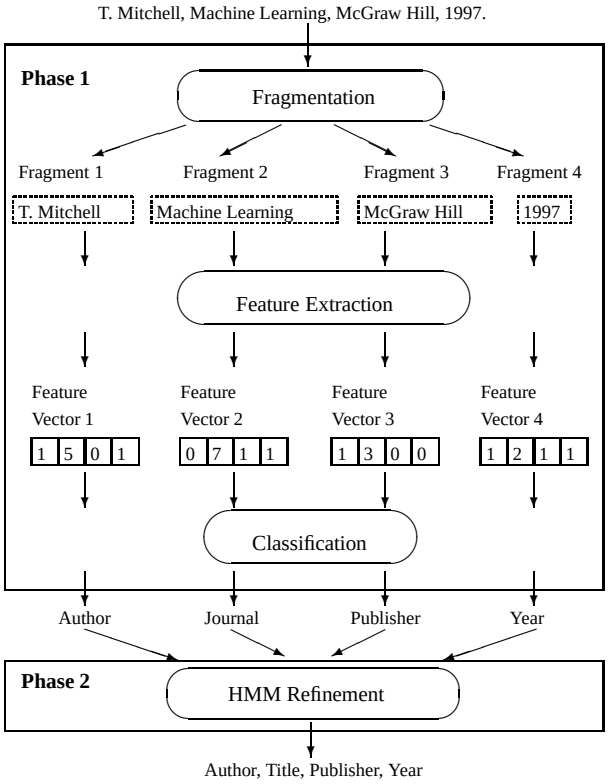

Figure 1. Proposed Approach

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}. 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}.

Each training example consists of a list of pairs containing a *symbol* (i.e., the class predicted to a specific fragment in Phase 1) and the associated *hidden state* (i.e., the class to which the fragment actually belongs). The transition probability $Pr[s'/s]$ and the emission probability $Pr[t/s]$ are estimated from a set of training sequences by using the following equations defined in [3]:

$$Pr[s'/s] = \frac{Number\ of\ transitions\ from\ s'\ to\ s}{Total\ number\ of\ transitions\ from\ s'} \quad (1)$$

$$Pr[t/s] = \frac{Number\ of\ emissions\ of\ t\ by\ state\ s}{Total\ number\ of\ symbols\ emitted\ by\ state\ s} \quad (2)$$

The HMM takes as input the whole sequence of class values provided by Phase 1 and returns a refined classification for the given fragments.

4 Case Study: IE on Bibliographic References

As case study, we chose the IE from bibliographic references aiming at the automatic creation of citation databases. It is possible to extract information from a reference, such as author(s), title, date of publication, etc. Bibliographic references are semi-structured texts with a high degree of variation in their structure [3]. The information to be extracted follows an ordering that, although not rigid, may help the extraction process. To take advantage of this structural ordering, the output delivered by Phase 1 is refined by an HMM.

4.1 Phase 1 - Extraction using a conventional text classifier

As seen above, Phase 1 is divided into three steps:

1. *Fragmentation of the input text*: This step was performed by using heuristics based on punctuation marks (e.g., ",", ".", ";", etc...). Here, for each punctuation mark found in the input text a new fragment is started. The only exception occurs when a punctuation is preceded by a single uppercase letter (since in our case study it commonly corresponds to name abbreviations).

2. *Feature extraction*: three distinct feature sets were used for describing the fragments: (1) Manual1 (20 features defined in [20]); (2) Manual2 (9 features defined in [4]); and (3) Automatic (100 words directly selected from the training corpus by Information Gain [30]). The first two sets were defined through knowledge engineering and contain features specific to the domain of references such as the occurrence of specific terms (e.g., "journal"), publisher names, etc.

3. *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. We used here four classifiers, implemented using the WEKA environment [28]: the Naive Bayes [13], the PART (Rules) algorithm [9], the k-Nearest Neighbour (k-NN) [1] and the Support Vector Machine (SVM) algorithm [5].

4.2 Phase 2 - Refinement of the Results Using an HMM

In this case study, the HMM structure was defined as follows: (1) it has one hidden state corresponding to each slot in the output form; and (2) all hidden states were connected to each other. In figure 2, we have a simplified example of HMM used in our case study with three hidden states.

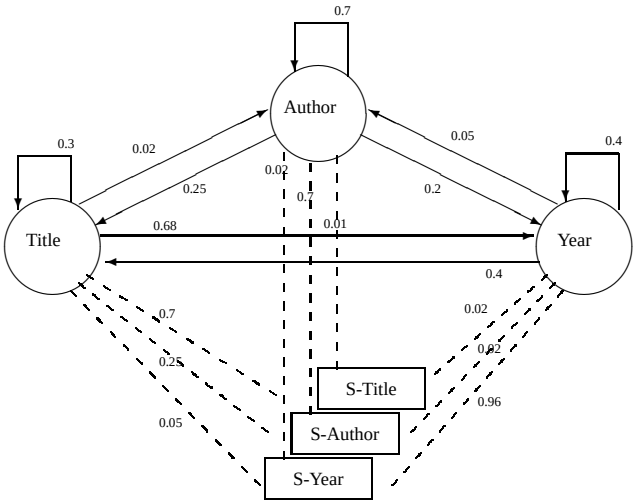

Figure 2. Example of HMM used in Phase 2

In order to implement this step, we deployed the libraries available in the BioJava project[2], which provides classes to define HMMs and to implement the Viterbi algorithm.

4.3 Experiments and Results

The prototype was evaluated using a corpus from a bibliography on computational linguistics[3] which contains 6000 references with tags that indicate the class of each text fragment. The experiments evaluated the performance of our system with HMM refinement compared to two different approaches for IE: the conventional text classification approach (Phase 1 of our system), and the Sliding Window approach. For that, the experiments were performed in three rounds:

1. First, we ran Phase 1 in isolation with different classifiers and feature sets (these test scenarios are described below). The aim was to have a basis to evaluate the quality of the refinement performed by the HMM.

2. Following, we ran the complete system (Phase 1 and 2) with the same test scenarios as above (i.e., varying classifiers and feature sets).

3. Finally, we performed experiments using the Sliding Window approach (see section 2). In this round, we aimed to evaluate our complete system against a

[2] BioJava (www.biojava.org) is an open-source project dedicated to providing a Java framework for processing data originally in the biological domain.

[3] Available in http://liinwww.ira.uka.de/bibliography/Ai/bateman.html

Table 1. Results obtained in the corpus of 6000 references. The best result for each combination of feature set and classifier is typed in boldface.

Feature Set	Classifier	Precision without HMM	Precision with HMM	Precision Sliding Window
Manual1	PART	72.36%	76.79%	**78.37%**
	Bayes	66.40%	**73.86%**	67.30%
	kNN	72.26%	**77.01%**	63.78%
	SVM	67.96%	**74.73%**	73.42%
Manual2	PART	77.59%	81.69%	**84.21%**
	Bayes	68.77%	**75.86%**	69.64%
	kNN	77.05%	**81.11%**	57.24%
	SVM	66.32%	**73.27%**	70.56%
Automatic	PART	51.51%	**69.65%**	36.25%
	Bayes	49.89%	**67.45%**	31.77%
	kNN	52.23%	**70.35%**	36.14%
	SVM	51.33%	**68.81%**	34.52%

benchmarking approach applied to IE in previous work [4]. In the experiments, we used an input window of width 3 (as adopted, for instance, in [4]).

The performance of the three evaluated approaches was estimated for 12 different scenarios (i.e., different combinations of feature set *versus* classifier). As seen in Section 4.1.1, we used four classifiers and three different feature sets. For each scenario, we applied a 10-fold cross-validation procedure to evaluate system's performance. The evaluation measure used was precision, defined as the number of correctly extracted slots divided by the total number of slots in the references. In order to verify statistical difference between the precision obtained by the evaluated approaches, we performed a paired t-test (see [17]).

Table 1 shows the average precision obtained by the three evaluated IE approaches for each combination of feature set versus classifier. By comparing the precision obtained with and without the HMM, we verify a gain in performance with the use of HMM in all combinations. The gain varied from 4.05 to 18.14 percentile points. By applying the paired t-test, we verified that the obtained gain was statistically significant at a 95% level of confidence in all combinations of feature sets and classifiers.

From Table 1, we can also observe that our system was in most settings of experiments better than the Sliding Window. In fact, on 10 of the 12 different combinations of feature set and classifier, the proposed system obtained a performance gain compared to Sliding Window. The difference between our method and the Sliding Window has shown to be statistically significant in all evaluated scenarios at a level of 95% of confidence.

Table 2. Average precision obtained by using different feature sets

Feature Set	Average Precision without HMM	Average Precision with HMM	Average Precision Sliding Window
Manual1	69.74%	75.60%	70.72%
Manual2	72.43%	77.98%	70.41%
Automatic	51.24%	69.07%	34.67%

Table 3. Average precision obtained by using different classifiers

Classifier	Average Precision without HMM	Average Precision with HMM	Average Precision Sliding Window
PART	67.15%	76.04%	66.27%
Bayes	61.68%	72.38%	56.23%
KNN	67.18%	76.15%	52.39%
SVM	61.86%	72.27%	59.49%

The IE performance was strongly influenced by the feature set used in the text classification (see Table 2). The best average results were obtained with the use of manual sets, which require a high level of expertise to be defined. The worst result was achieved by using the Automatic set, which represents the less expressive set. However, the use of HMMs improved the low performance of the Automatic set, delivering final results closer to those obtained with the other feature sets. The HMM is able to compensate the use of a less expressive feature set, thus facilitating the customization of the system to different IE domains.

Table 3 shows that the IE performance was also influenced by the algorithm used as text classifier. However, with the use of HMMs, the difference in performance between the best and the worst algorithms (3.88 percentile points) was lower when compared to the results observed by using the other IE approaches. The variability of the system performance, regarding the classifier used in Phase 1, is lower when the HMM is used in the refinement phase.

5 Conclusion

In this work, we presented a hybrid machine learning approach for IE which combines text classifiers and Hidden Markov Models. In order to evaluate the viability of our proposal, we implemented a prototype and performed experiments in the task of IE on bibliographic references. The experiments on a corpus of 6000 references have revealed a significant improvement in performance when the HMM is used to refine the outputs of the text classifiers.

The performed experiments also showed that the use of

an HMM compensated the low performance of less adequate classifiers and feature sets. A high precision average was obtained even with features defined without an expert's effort. The variability in performance, considering the algorithm used as text classifier, was also lower compared to the other evaluated IE approaches.

Despite the satisfactory results obtained in our proposal, it still has some limitations which will be dealt with in future work. Currently, all hidden states are connected to each other, which is a naive strategy to define the HMM structure. In future work, the HMM structure can be defined by using optimization techniques (such as genetic algorithms). We also intend to use more sophisticated techniques in the fragmentation of the input documents, and to evaluate the impact of this step in the whole extraction process. Finally, we intend to evaluate the proposed approach to other IE tasks.

References

[1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6(3):37–66, 1991.

[2] R. Baeza-Yates and B. R. Neto. *Modern Information Retrieval*. Addisonn-Wesley, New York, EUA, 1999.

[3] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 175–186, 2001.

[4] R. R. Bouckaert. Low level information extraction: a bayesian network based approach. In *TextML*, 2002.

[5] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[6] H. Chieu and H. Ng. A maximum entropy approach to information extraction from semistructured and free text. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, pages 786–791, 2002.

[7] T. G. Dietterich. Machine learning for sequential data - a review. *Lecture Notes in Computer Science*, 2396:15–30, 2002.

[8] R. Feldman, B. Rosenfeld, and M. Fresko. Teg a hybrid approach to information extraction. *Knowledge and Information Systems*, 9(1):1–18, 2006.

[9] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the 15th Intern. Conf. on Machine Learning*, pages 144–151, 1998.

[10] D. Freitag. *Machine learning for information extraction in informal domains*. PhD thesis, Computer Science Department, Carnegie Mellon University, 1998.

[11] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the ECAI Workshop on Machine Learning for Information Extraction*, 2000.

[12] C. Giraud-Carrier, R. Vilalta, and P. Brazdil. Introduction to the special issue on meta-learning. *Machine Learning*, 54(3):187–193, 2004.

[13] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proc. of the 11th Confer. on Uncertainty in Artificial Intelligence*, pages 338–345, 1995.

[14] R. Kosala, V. den Bussche, M. Bruynooghe, and H. Blockeel. Information extraction in structured documents using tree automata induction. In *Proc. of the 6th PKDD*, 2002.

[15] N. Kushmerick, E. Johnston, and S. McGuinness. Information extraction by text classification. In *IJCAI-01 Workshop on Adaptive Text Extraction and Mining*, 2001.

[16] M. C. M. Berardi and D. Malerba. A hybrid strategy for knowledge extraction from biomedical documents. In *ICDAR workshop on Neural Networks and Learning in Document Analysis and Recognition*, 2005.

[17] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[18] I. Muslea, S. Minton, and C. Knoblock. Active learning with multiple views. *Journal of Artificial Intelligence Research*, 27:203–233, 2006.

[19] G. Neumann. A hybrid machine learning approach for information extraction from free text. In *Proc. of the 29th Annual Conf. of the German Classification Society*, 2005.

[20] C. Nunes and F. A. Barros. Prodext - a knowledge-based wrapper for extraction of technical and scientific production in web pages. In *Proceedings of the International Joint Conference IBERAMIA-SBIA 2000*, pages 106–115, 2000.

[21] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[22] G. Sigletos, G. Paliouras, and C. Spyropoulos. Combining information extraction systems using voting and stacked generalization. *Journal of Machine Learning Research*, 6:1751–1782, 2005.

[23] E. Silva, F. Barros, and R. Prudêncio. A hybrid machine learning approach for information extraction. In *Proceedings of the 6th International Conference on Hybrid Intelligent Systems*, 2006.

[24] A. D. Sitter and W. Daelemans. Information extraction via double classification. In *Proc. of the Intern. Workshop on Adaptive Text Extraction and Mining*, pages 66–73, 2003.

[25] M. Skounakis, M. Craven, and S. Ray. Hierarchical hidden markov models for information extraction. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.

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

[27] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Computing Surveys*, 38(2), 2006.

[28] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[29] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[30] Y. Yang and J. O. Pedersen. A comparative study on feature selection methods in text categorization. In *Procceedings of the 14th ICML*, pages 412–420, 1997.

[31] J. Zavrel, P. Berck, and W. Lavrijssen. Information extraction by text classification: Corpus mining for features. In *Proceedings of the workshop Information Extraction meets Corpus Linguistics*, 2000.