RecBoost: A Supervised Approach to Schema Reconciliation

Eugenio Cesario, Francesco Folino, Antonio Locane, Giuseppe Manco, and Riccardo Ortale

ICAR-CNR
Via Bucci 41c
I87036 Rende (CS)
Italy
{surname}@icar.cnr.it

Abstract. In several applicative scenarios, large quantities of data are collected and stored as free text. In many cases, the information stored has a latent schema consisting of a set of attributes, that would in principle allow to fit such textual data into an appropriate relational structure. In this paper, a novel approach for bringing to the surface the implicit attribute schema is proposed. The basic idea is to subject the available text to a *progressive classification*, i.e., a multi-stage classification scheme where, at each intermediate stage, a classifier is learnt that analyzes the textual fragments not reconciled at the end of the previous steps. Classification is accomplished by ad-hoc exploitation of traditional association mining algorithms, and is supported by a data transformation scheme which takes advantage of domain-specific dictionaries/ontologies. A key feature is the capability of progressively enriching the available ontology with the results of the previous stages of classification, thus significantly improving the overall classification accuracy. An extensive experimental evaluation shows the effectiveness of our approach on real datasets.

1 Introduction

The wide exploitation of new techniques and systems for generating, collecting and storing data has made available a huge amount of information. Large quantities of such data are stored as continuous text. In many cases, this information has a latent schema consisting of a set of attributes, that would in principle allow to fit such textual data into some field structure, so that to exploit the mature relational technology for more effective information management. For instance, personal demographic information typically comprises names, addresses, zip codes and place names, which indicate a convenient organization for the these kind of data. However, the extraction of structure from textual data poses several challenging issues, since free text does not necessarily exhibit a uniform representation.

Foremost, the order of appearance of the attributes across the individual lines of text may not be fixed. In addition, their recognition is further complicated by the absence of both suitable field separators and a canonical encoding format, which is mainly due to erroneous data-entry, misspelled terms, transposition oversights, inconsistent data collection and so forth. As a concrete example, common issues in personal demographic data are the adoption of abbreviations for both proper names and common terms and the availability of multiple schemes for formatting addresses, phone numbers and birth dates.

Also, distinct records may lack different attribute values, which makes them appear with a variable structure. Yet, the same data may be fragmented over disparate data sources, which further exacerbates the aforementioned difficulties.

The notion of *Entity Resolution* [11, 9, 23],denotes a complex process for database manipulation that embraces three primary tasks. *Schema reconciliation* consists in the identification of a common field structure for the information in a data source. *Data reconciliation* is the act of discovering synonymies in the data, i.e. apparently different records that, as a matter of fact, refer to a same real-world entity. *Identity definition* groups tuples previously discovered as synonymies, and extracts a representative tuple for each discovered group.

In this paper, we propose RecBoost, a novel approach to schema reconciliation, that adopts classification as an effective mechanism for fragmenting free text into tuples with a common attributes structure. RecBoost works by performing two macro-steps, namely preprocessing and reconciliation. The former step is primarily thought for formatting the individual lines of text, with potentially-different encoding format, into a uniform representation. Domain-specific ontologies and dictionaries are then exploited to associate each such a token with a label denoting its ontological or syntactic category.

Reconciliation is eventually accomplished in terms of *progressive classification*, i.e., a multi-stage classification scheme where, at each intermediate stage, a classifier is learnt from the previous classification outcome, by analyzing the textual fragments not reconciled yet.

The main contributions of our study are summarized next.

- The joint exploitation of ontology and dictionary-based generalization with rule-based classification, which allows to reliably associate terms in a free text with a corresponding semantic category. This allows to fit the textual information into a proper structure with fields for each such a category.
- The introduction of *progressive classification*, as an effective mechanism for achieving an accurate schema reconciliation. At each stage of the overall classification process, a suitable rule-based classifier is learnt from the outcome of the previous stage, thus being specifically targeted at handling with those terms that have not yet been associated with any semantic category.
- An intensive experimental evaluation on real-world data, that investigates and confirms the effectiveness of our approach from several facets.

The outline of the paper is as follows. Section 2 introduces the basic notation for the problem we face, and overviews works from the literature, that are most closely related to our study. Section 3 begins by covering details on the process adopted to learn a generic rule-based classifier and, then, proceeds to examine the RecBoost methodology. This is hence exemplified in section 5. In section 6 the results of an intensive experimental evaluation are presented. Finally, section 7 draws some conclusions and highlights a number of interesting directions, that are worth further research.

2 Background and Related Work

To formalize the Schema Reconciliation problem, we assume the following basic definitions. An *item domain* $\mathcal{M} = \{a_1, a_2, \ldots, a_M\}$ is a collection of *items*. Let s be a *sequence* $a_1, \ldots, a_m$ where $a_i \in \mathcal{M}$. The set of all possible sequences is denoted by $\mathcal{M}^*$. In general, an item a_k belongs to a sequence s (denoted by $a_k \in s$) if $s = a_1, a_2, \ldots, a_k, \ldots, a_n$. Moreover, we denote the subsequence $a_1, a_2, \ldots, a_{k-1}$ as $pre_s(a_k)$, and the subsequence $a_{k+1}, a_{k+2}, \ldots, a_n$ as $post_s(a_k)$.

A *descriptor* $R = \{A_1, \ldots, A_n\}$ is a set of labels. A descriptor corresponds to a database schema, with the simplification that, for each attribute label A_i, has the same domain $\mathcal{M}^*$ associated. Given a *descriptor* $R = \{A_1, \ldots, A_n\}$, a tuple τ_R is defined as $\{A_1 : s_1; A_2 : s_2; \ldots; A_n : s_n\}$, where s_i is a sequence. As an example, given the descriptor $R = \{\text{NAME}, \text{ADRRESS}, \text{CITY}\}$, a tuple for such descriptor is

$$\begin{array}{l}\{\ \text{NAME} : \texttt{Alfred Whilem};\\ \ \ \text{ADDRESS} : \texttt{Salisbury Hill, 3001};\\ \ \ \text{CITY} : \texttt{London} \quad\ \}\end{array}$$

Thus, our specific problem can be viewed as follows: given a descriptor (database relation) $R = \{A_1, \ldots, A_n\}$, and a data set of sequences (free text) $S = \{s_1, \ldots, s_m\}$, we want to segment each sequence s_i into subsequences $s_i^1, \ldots, s_i^k$, such that each token $a \in s_i^h$ is associated with the proper attribute A_j.

For example we may want to fit an unstructured collection of personal demographic information representing names, addresses, zip codes and places, in a proper schema with specific fields for each category, as shown in the figures below.

s_1	Harry Hacker Northern Boulevard 3001 London
s_2	C Cracker Salisbury Hill Flushing
s_3	Tony Tester Brooklyn Johnson Avenue 2

	NAME	ADDRESS	ZIP CODE	CITY
s_1	Harry Hacker	Northern Boulevard	3001	London
s_2	C Cracker	Salisbury Hill		Flushing
s_3	Tony Tester	Johnson Avenue	2	Brooklyn

The problem of harmonizing postal addresses affects large organizations like banks, telephone companies and universities, which typically collect millions of unformatted address records. Since each address record can in principle be retrieved from a different data source (designed with different purposes), variations in the way such records are stored are far from unusual. Besides the above described, there are

several further applicative scenarios requiring to takle the schema reconciliation problem. These include bibliographic records, collections of information about products, medical sheets, etc.

The problem is clearly related with *Part Of Speech* (POS) *Tagging* and Shallow Parsing, *Wrapping* and, in general, with the problem of extracting structure from free text. The aim of POS Tagging is to assign labels to speech words that reflect their syntactic category. To this purpose, both statistical [14, 17, 4, 22, 12] and rule-based methods [6, 22, 15, 7] have been proposed in the literature. In practice, the basic idea behind POS tagging consists in disambiguating phrases by exploiting dictionaries and analyzing the text context surrounding the candidate entity. However, the approach fails at treating "exceptions", i.e., words that are not included in a dictionary, such as proper names, cities, or addresses. By contrast, these are exactly the features which characterize our scenario.

As far as wrapping is concerned, most algorithms considerably rely on HTML separator tags, and on the fact that data represent a regular multi-attribute list [10]. Such approaches are not effective in domains where data do not necessarily adhere to a fixed schema. Indeed, instances in our problem are more irregular -the order of fields is not fixed, not all attributes are present, etc. The classification of an item is better performed according to its neighboring words, absolute/relative position in the string, numeric/alphanumeric characters, and so on. To our knowledge, few exception are capable of effectively dealing with such features [20, 1, 8]. For example, WHISK (proposed in [20]) can deal with missing values and permutation of fields, but it requires a "complete" training set, i.e. a set of examples including all the possible occurrences of values.

Several recent approaches to schema reconciliation rely on Hidden Markov Models (HMM) [2, 5, 18, 16]. A HMM [19] consists of a set of states and pointed edges among such states. Two particular states are the *initial* and the *final* states: The former has no incoming edges, whereas the latter has no outgoing edges. Each state of the HMM, with the exception of the *initial* and *final* ones, represents a class-label (an attribute of the relational schema) and is associated with a set of emission probabilities. Precisely, each state is associated with as many emission probabilities as terms in a dictionary containing all the terms encountered in the training phase. Moreover, edges among states are associated with transition probabilities.

Schema reconciliation with HMM can be accomplished by learning the structure of a HMM, and applying the discovered structure to unknown examples. For example, DATAMOLD [5] achieves classification by associating each term with all possible states, according to the corresponding emission probabilities. Since a term may be associated with more than one state, each sequence can in principle be mapped with multiple paths in the HMM. A path is any edge-connected route between the *initial* and the *final* states. Transition and emission probabilities are then exploited to evaluate the most likely path among the possible ones, and the terms are then classified accordingly.

The effectiveness of the approaches based on HMM strongly depends on the number of terms appearing in the training set. Furthermore, there is a significant difference with regard to the approach proposed in this paper. Indeed, a HMM classifies each sequence of terms in one step, whereas our approach foresees a segmentation of each sequence into tokens, and then a classification token by token. The latter eases the application of a pipeline of classifiers, which allow the progressively increase the classification accuracy on unknown terms.

Specifically, we employ a variant of the *Apriori* [3] algorithm to find associations among tokens, from which to filter suitable classification rules. This latter task is attained by adopting a variant of the *Classification Based on Association - Classifier Builder (CBA-CB)* algorithms [13]. The main advantage of splitting the overall classification process into various stages consists in pursuing a *progressive classification*. Practically, a specific classifier is adopted at each stage, which attempts at handling with those tokens that remained unclassified at the end of the previous step.

3 The RecBoost Methodology

The reconciliation of a set $S = \{s_1, \ldots, s_m\}$ of sequences with an attribute schema $R = \{A_1, \ldots, A_n\}$ consists in the association of each token a within the generic sequence $s \in S$ with an appropriate attribute of R.

RecBoost pursues text reconciliation via term generalization. Precisely, two types of generalizations are involved, namely syntactic and ontological analysis, and contextual generalization. The former aims at labelling textual tokens with their syntactic or ontological categories. The latter employs knowledge of the relationships among textual tokens, ontological categories and schema attributes, for assigning each token to a proper schema attribute.

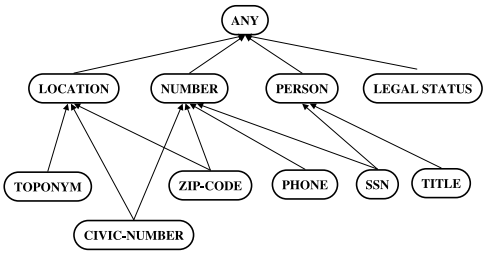


Fig. 1. An example concept hierarchy

As an example, a token a composed by multiple consecutive digits may be ontologically denoted as a number. Subsequently, the contextual presence on the same sequence containing a of the further ontological labels city and street (which directly follow and precede a), may determine the reconciliation of a with an attribute address of the schema descriptor.

Subsections 3.1 and 3.2 respectively delve into the details of syntactic and contextual analysis.

3.1 Syntactic and Ontological Analysis

RecBoost exploits a user-defined domain ontology in order to preprocess sequences within S. In practice, a domain ontology is specified as $\mathcal{G} = \langle \mathcal{L}, \lhd, \mathcal{A} \rangle$, where

- $\mathcal{L}$ is a set of categories,
- $\lhd$ is a precedence relation defined on $\mathcal{L}$, and
- $\mathcal{A}$ is a set of rules of the form

if *Condition*
then *Action*

Intuitively, $\mathcal{L}$ represents a set of ontological concepts, which can be exploited in order to generalize tokens within a sequence. Such concepts are structured in a concept hierarchy, specified by the $\lhd$ relation. Figure 1 exemplifies a set of concepts and their hierarchical relation. For example, the concept NUMBER is related to 3 subconcepts, namely ZIP-CODE, PHONE and CIVIC-NUMBER.

Rules in $\mathcal{A}$ are useful to specify background knowledge about the domain under consideration, and are meant to provide a transformation of a set of tokens appearing in a sequence. Specifically, *Condition* specifies a pattern-matching expression defined over the tokens of a sequence, and *Action* specifies some actions to take on such tokens. We consider two main actions, here. The first is a *labeling action*, which substitute a token (or a set of tokens) with a concept in $\mathcal{L}$. An example rule specifying such an action is

r_1: **if** a is a four-digits token
then **replace** a with ZIP-CODE

We can also specify *restructuring actions*, which operate on a set of tokens by applying basic transformation operations (such as deleting, merging or segmenting). An example is

r_2: **if** a_i is a four-digits token
and a_{i+1} is a token containing digits
then **merge** a_i and a_{i+1} into a new token a

We also assume that some rules can be supported by user-defined dictionaries. For example, the rule

r_3: **if** $a \in$ DICTIONARY
then **replace** a with TOPONYM

specifies that each token appearing in the set DICTIONARY of all known toponyms (such as `street`, `road`, `blvd`, and so on) can be generalized with the category TOPONYM in $\mathcal{L}$.

By exploiting $\mathcal{G}$, syntactic generalization performs two steps. First, it transforms the original sequences in $S = \{s_1, \ldots, s_n\}$ into a new set $S' = \{s'_1, \ldots, s'_n\}$, where each sequence s'_i is obtained from s_i by

applying the rules in $\mathcal{A}$. [1] Second, the available tokens in each sequence are further generalized by an ad-hoc exploitation of the hierarchy described by the $\triangleleft$ relation. The exploitation is a direct result of a cooperation with contextual analysys, which reconciles tokens in S' as described in the next subsection.

3.2 Contextual Analysis

This steps is meant to associate tokens in S with their corresponding attribute in R. We approach the problem from a supervised learning perspective. Formally, we assume that there exists a function $\lambda : \mathcal{M}^* \mapsto \mathcal{M} \mapsto R$ that, for each sequence $s \in \mathcal{M}^*$, labels a token a into a schema attribute A_j, namely $\lambda_s(a) = A_j \in R$. Hence, the problem can be stated as learning λ from a training set T such that, for each sequence $s \in T$ and for each token $a_i \in s$, the label $\lambda_s(a_i)$ is known.

In order to correctly classify each token $a_i \in s$, information about its *context* is needed. The "context" of a generic token $a_i \in s$, is the set of all the items preceding and following a_i in s. Thus, we hold the context of a_i introducing the notation:

$$feature_s(a_i) = \langle pre_s(a_i),\ a_i\ , post_s(a_i)\rangle$$

The set $\mathcal{T} = \{\langle feature_s(a), \lambda_s(a)\rangle | s \in T, a \in s\}$ represents the training set for our classification problem.

The idea beyond contextual analysis is to examine the context $feature_s(a)$ of a each token a within any sequence s, in order to learn meaningful associations among groups of tokens of S. These associations can be then exploited to learn a rule-based classifier, that associates each individual token in S with an attribute in R.

In practice, our objective is to build a classifier $C : \mathcal{M} \mapsto (\mathcal{M} \cup \mathcal{L} \cup R)^* \mapsto R$, specified by rules of the form

if *Condition*
then $\lambda_s(a)$=*Class*

Here, a and s represent, respectively, token and sequence variables. Moreover, *Condition* represents a conjunction of terms, and *Class* represents an attribute in R. Terms in *Condition* can be specified in three different forms: either as $a = v$, $v \in pre_s(a)$ or $v \in post_s(a)$, where v is any constant in $\mathcal{M} \cup \mathcal{L} \cup R$.

Learning a Rule-based Classifier We here discuss the process of distilling a rule-based classifier from a training set T, for which the correspondences between tokens in T and labels are known. A holdout approach is adopted to partition T into a validation set V and an actual training set $D = T - V$. The goal is learning a classifier from D that has highest accuracy on V.

In principle, any rule-based classifier could be used here. However, we found that classification based on association rules is more effective in this setting than, e.g., traditional algorithms based on decision-tree learning. The intuition behind the above statement is that association rules are better suited to detect local patterns which hold "locally" on small subsets of D. This is especially true when D is large, and contains many contrasting specificities across individual sequences. By contrast, decision trees represent global models, which are hardly able to capture such specificities without incurring into the overfitting phenomenon. In addition, the intrinsic unstructured nature of the feature space to analyze does not allow an immediate application of decision-tree learning techniques, whereas association rules mining techniques naturally fit the domains under consideration.

A variant of the Apriori algorithm [3] is exploited to extract from the explicit representation of token contexts, $\mathcal{D} = \{\langle feature_s(a), A\rangle | s \in D, a \in s, A \in R\}$, a set of association rules that meet pre-specified requirements on their support and confidence values and whose consequents are narrowed to individual schema attributes. A classifier can hence be built on the basis of such discovered rules, by selecting the most promising subset, i.e, the subset of rules which guarantees the maximal accuracy. To this purpose, we adopted the CBA-CB method [13], which allows an effective heuristic search for the most accurate association rules. Its basic idea is succinctly elucidated next.

A precedence-operator is exploited as a total order among the association rules in $\mathcal{D}$. Formally, given any two rules r_i and r_j, r_i is said to have a higher precedence than r_j, which is denoted by $r_i \prec r_j$, if:

[1] Notice that, since multiple matching preconditions can hold for the same set of tokens, rules in $\mathcal{A}$ are applied in a user-defined order. In the above example, r_2 has a precedence on r_1, since in principle a token containing 4 digits can be interpreted as a zip code if and only if it is not followed by a new number (in which case is has to be interpreted as an area code in a phone number).

```
1:  Classifier := ∅;
2:  CARs := sort(CARs);
3:  for all r ∈ CARs do
4:     temp := ∅;
5:     marked := false;
6:     for all t ∈ T do
7:        if t satisfies conditions of r then
8:           temp := temp ∪{t}
9:           marked := true
10:       end if
11:    end for
12:    if marked then
13:       insert r at the end of Classifier;
14:       delete all cases in temp from T;
15:    end if
16: end for
```

Fig. 2. A naive scheme for learning a rule-based classifier

1. the confidence of r_i is greater than that of r_j, or
2. their confidences are the same, but the support of r_i is greater than that of r_j, or
3. both confidences and supports are the same, but the antecedent of r_i is shorter than r_j.

Hence, a classifier can be formed by choosing a set of high precedence rules to cover $\mathcal{D}$. The resulting classifier can be modelled as:

$$\langle r_1, r_2, \ldots, r_n \rangle$$

where $r_i \in \mathcal{D}$, $r_a \prec r_b \; if \; b > a$. While considering an unseen case of $\mathcal{D}$, the first rule that covers the case also classifies it. Clearly, if no rule applies to a given case, the case is unclassified.

The CBA-CB approach assumes that:

1. each case in the training set $\mathcal{D}$ is covered by the rule with the highest precedence among those that can actually cover the case;
2. every rule in the classifier correctly classifies at least one remaining case in $\mathcal{D}$, when it is chosen.

In principle, the fulfillment of both conditions can be guaranteed by a naive search scheme such as the one in fig. 2. The algorithm starts (row 2) by sorting all the rules in $\mathcal{D}$ by their mutual precedence. This guarantees the fulfilment of condition 1. Hence, the classifier is built by progressively selecting rules from the resulting sequence of sorted rules. A temporary set *temp* (row 4) and a boolean value *marked* (row 5) are exploited to the purpose of processing each rule r in the sequence. Precisely, the idea is to find cases in $\mathcal{D}$, that are covered by r and to accordingly update both *temp* and *marked* (row 6-11). For each rule r that correctly classifies at least one case two actions are performed: the insertion of r at the end of the classifier and the consequent deletion of the cases covered by r itself from $\mathcal{D}$ (row 12-15). Clearly, rules that classify no cases are discarded.

We further acted on the scheme shown in fig. 2 by implementing a post-processing strategy, which aims at further improving the classification accuracy of the discovered rules. The postprocessing is mainly composed by attribute and rule pruning. The idea behind attribute pruning consists in removing items from classification rules, whenever this does not worsen the error rate of the resulting classifier. The validation set V is exploited to asses classification accuracy. Precisely, let r be a generic classification rule containing at least two terms in the antecedent. Also, assume that s denotes a generic sequence in V and that x represents a token within s. The error r_x of rule r on x is a random variable

$$r_x = \begin{cases} 1 \text{ if } r \text{ misclassifies } x \\ 0 \text{ otherwise} \end{cases}$$

Hence, the overall error of r on V can be defined as follows

$$E(r) = \frac{1}{n_V} \sum_{x,s/x \in s, s \in V} r_x$$

where n_V indicates the overall number of tokens within V. A new rule r' can now be generated by removing from the antecedent of r any of its terms. We replace r by r' if two conditions hold, namely $E(r') < E(r)$ and the discrepancy $E(r) - E(r)'$ is statistically relevant. To verify this latter condition, we exploit the fact that for n_V large, the distribution of $E(r)$ approaches the normal distribution. Hence, we compute a $\tau\%$ confidence interval $[\alpha, \beta]$, whose lower and upper bounds are respectively given by

$$\alpha = E(r) - c_\tau \sqrt{\frac{E(r)[1 - E(r)]}{n_V}}$$

and

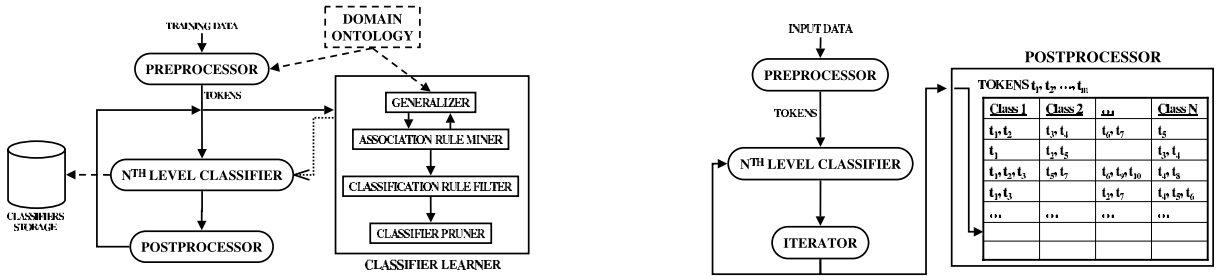
$$\beta = E(r) + c_\tau \sqrt{\frac{E(r)[1 - E(r)]}{n_V}}$$

where, constant c_τ depends on the confidence threshold τ. The above interval represents an estimate for the actual error of rule r.

Finally, we retain r' instead of r, if it holds that $E(r') < \alpha$. In such a case, we analogously proceed to attempt at pruning further items from the antecedent of r'. Otherwise, we reject r'.

Rule pruning instead aims at reducing the number of rules in a classifier. As in the case of attribute pruning, the idea consists in removing rules from a classifier, whenever this does not worsen the accuracy (on some test set) of the resulting classifier. To this purpose, all rules in a classifier are individually evaluated on the basis of their precedence order. A generic rule r is removed, if one of the following conditions holds:

- r does not cover a minimum number of cases in V;
- the accuracy of r on V is below a minimum threshold;
- the removal of r from the classifier increases its overall accuracy on V.



(a) Training (b) Classification

Fig. 3. Training and Classification phases in the RecBoost methodology.

4 RecBoost Anatomy

Association rules for classification allow to tune the underlying classification model to a local sensitivity. However, in principle their adoption can yield a high number of unclassified tokens. In a reconciliation scenario, this is due to the presence of unknown or rare tokens, as well as errors in the text to segment. The adoption of concept hierarchy alleviates such an effect -and indeed it has already been adopted in traditional approaches based on HMM [2, 5]. The novelty in the RecBoost reconciliation methodology relies on a finer cooperation between synthactic/ontological analysis and contextual analysis. The reiteration of the process of transforming tokens and learning a rule-based classifier allows *progressive classification*, i.e., the adoption of multiple stages of classification for more effective text reconciliation. Precisely, a pipeline $\mathcal{C} = \{C_1, \ldots, C_k\}$ of rule-based classifiers is exploited to this purpose. Here, the generic classifier C_i, $i = 2, \ldots, k$, is specifically learnt, as discussed in subsection 3.2, to classify all those tokens, that were not reconciliated at the end of step $i - 1$. The length k of the classification pipeline is chosen in order to achieve an optimal overall classification accuracy.

Let s be a sequence of S and a a token within s. The behaviour of C_i can be formally described in terms of a partial mapping $C_i = (\mathcal{M} \times \mathcal{G} \cup R)^* \mapsto \mathcal{M} \mapsto R$. The behavior of C_i relies on a specific training set T_i, that is obtained from T_{i-1} by adding domain information as resulting from the classifier C_{i-1}. Given a sequence $s \in T_{i-1}$, the idea is evaluating the set X_s of tokens in s, that are not covered by any rule of C_{i-1}, by enriching the domain information in $\mathcal{G}$ with a new set of rules directly extracted from the set of classification rules in C_{i-1}. Specifically, each classification rule $r \in C_{i-1}$

if *Condition*
then $\lambda_s(a)$=*Class*

is transformed into a labeling rule r'

if *Condition*
then **replace** a with *Class*

and the new rule r' is added to the set $\mathcal{A}$ of rules available for syntactic analysis. Then, syntactic analysis is applied to each sequence s in T_{i-1}, and the resulting transformed sequences are collected in T_i. A new training set $\mathcal{T}_i$ is then generated by collecting, for each sequence $s \in T_i$ and each token $a \in X_s$, the tuples $\langle feature_s(a), \lambda_s(a) \rangle$. Notice that there is a direct correspondence between the context $feature_s(a)$ computed at step i and the context computed at step $i-1$. Indeed, the new context $feature_s(a)$ follows from the context of a within T_{i-1} by replacing each token $b \notin X_s$ of s with its corresponding attribute $C_{i-1}(b)$.

The above detailed methodology is supported by three main components, namely a *preprocessor* (*tokenizer*), a *classifier learner* and a *postprocessor*. The components cooperate both in the training and in the classification phases, as detailed in fig. 3. In the following, we explain the role played by each of the aforementioned modules in the methodology. Section 5 provides a running example that better elucidates the overall classification process.

4.1 Preprocessor

A cleaning step is initially performed by this component, to the purpose of encoding the initial data sequences of a free text S into a uniform representation. This phases involves typical text processing operations, such as the removal of stop-words, extra blank spaces, superfluous hyphens and so forth.

The *preprocessor* then proceeds to split free text into tokens. The main goal of this phase is to recognize domain-dependent symbol-aggregates (e.g. acronyms, telephone numbers, phrasal construction, and so on) as single tokens. As an example, aggregates such as '*I B M*', '*G. m. b. H.*' or '*as well as*' are more significant as single units, rather than as sequences of words in the text. The identification of symbol aggregates as well as domain/specific cleaning steps are accomplished by using domain-specific transformation rules suitably defined in $\mathcal{G}$.

4.2 Classifier Learner

The *classifier learner* is responsible for producing an optimal set of classification rules, as shown in fig. 3(a). It consists of four main elements: a *generalizer*, an *association rule miner*, a filter for *classification rules* and a *classifier pruner*.

The *generalizer* performs ontological generalization, by exploiting the labeling rules and the $\triangleleft$ relationship defined in $\mathcal{G}$. Its role is mainly to enable the discovery of accurate association/classification rules, by providing an adequate degree of generalization among the data which effectively support the generation of rules frequent and not trivial. To accomplish this task, the generalizer enables the labeling rules in $\mathcal{A}$. Next, for each label replacing a token in the sequence, the related concept hierarchy is inspected, and an adequate degree of generalization for it is detected. The latter operation is performed in cooperation with the *association rule miner*, which implements the mining strategy described in [21] and is is targeted at extracting a complete set of association rules on the basis of given constraints on their minimum *support* and *confidence*.

Finally, the *classification-rule filter* distillates a classifier from the discovered association rules, and the *classifier pruner* attempts to reduce the overall size of a classifier by means of techniques for attribute and rule pruning. These techniques are detailed in section 3.2.

4.3 Postprocessor

The *postprocessor* rebuilds the sequences reconciled by a rule-based classifier, at any stage of progressive classification, by fitting them into a relational structure with schema R, as shown in fig. 3(b). This is accomplished by interpreting each (partially) reconciled sequence as a structured tuple, and organizing the tokens that have been so far reconciled as a collection of values for their corresponding schema attributes. Postprocessing enables progressive reconciliation: at any stage, a classifier is specifically learnt for dealing with those sequence tokens, that were not reconciliated at the end of the previous stage.

The postprocessor is also used in support of the training phase, as shown in fig. 3(a). There, its main role is to prepare the i-th training set T_i by generalizing the tokens in each sequence $s \in T_{i-1}$ via the application of the rules in C_{i-1}.

5 An Illustrative Example

PRE	WORD	POST
-	Harry	Hacker TOPONYM Boulevard ZIP London
Harry	Hacker	TOPONYM Boulevard ZIP London
Harry Hacker	TOPONYM	Boulevard ZIP London
Harry Hacker TOPONYM	Boulevard	ZIP London
Harry Hacker TOPONYM Boulevard	ZIP	London
Harry Hacker TOPONYM Boulevard ZIP	London	-

Fig. 4. Pre-word-post representation

We elucidate the overall RecBoost methodology, by exemplifying the reconciliation of a collection of personal demographic information, shown in the figure below, in compliance with the attribute descriptor $R = \{NAME, ADDRESS, ZIP, CITY\}$.

s_1	Harry Hacker 348.2598781 "Northern - Boulevard" (3001) London
s_2	C. Cracker ... Salisbury Hill, Flushing
s_3	Tony Tester Johnson Avenue 2 -Brooklyn- 323-45-4532

In particular, we assume to exploit a dictionary D, containing all known toponyms, and a domain-specific ontology $\mathcal{G} = \langle \mathcal{L}, \lhd, \mathcal{A} \rangle$, such that $\mathcal{A}$ consists of the following ontological rules:

r_1: **if** a is a four-digits token
then **replace** a with ZIP-CODE

r_2: **if** a is a token of more that four digits
then **replace** a with PHONE-NUMBER

r_3: **if** a is a token of type $ddd - dd - dddd$,
and d is a digit
then **replace** a with SSN

r_4: **if** $a \in$ DICTIONARY
then **replace** a with TOPONYM

The example data collection is corrupted by noise, i.e. by the absence of a uniform representation for all of its constituting sequences. Indeed, a comparative analysis of their formatting encodings reveals that:

- there is a telephone number in sequence s_1 that has to be discarded, since it is not expected by the descriptor R;

– character '-' is employed in sequence s_1 as separator between the words Northern and Boulevard, that are in turn delimited by double quotes;
– brackets are exploited to separate the zip-code information in sequence s_1;
– three non-relevant dots precede the address information in sequence s_2;
– two hyphens in sequence s_3 demarcate the word Brooklyn;
– there is a social security number (SSN) in sequence s_3 that has to be discarded, since it is not expected by the descriptor R.

The identification of a uniform representation format for all the individual sequences in the textual database enables an effective segmentation of such sequences into tokens and, hence, a reliable reconciliation. A preprocessing step is performed to this purpose.

5.1 Preprocessing

At this step, spurious characters are removed from the available data sequences, to the purpose of formatting them into a same encoding style. The resulting sequences are subsequently segmented into meaningful tokens. The output of this step is represented in the figure below.

s_1	Harry	Hacker	3482598781	Northern	Boulevard	3001	London
s_2	C	Cracker	Salisbury	Hill	Flushing		
s_3	Tony	Tester	Johnson	Avenue	2	Brooklyn	323-45-4532

The fragmented text is now subjected to a pipeline of rule-based classifiers, that reconciliate groups of tokens across the individual sequences s_1, s_2, s_3 with the attributes in R.

For the sake of convenience, we assume that two stages of classification allow the accomplishment of an actual reconciliation. Furthermore, since progressive classification involves a similar processing for each sequence in the tokenized text, we proceed to exemplify the sole reconciliation of s_1.

5.2 Progressive Classification

Progressive classification divides into syntactic and contextual analysis.

Syntactic Analysis This step performs token generalization. Here, the exploitation of the above ontological rules allow the generalization of a number of tokens in s_1 as shown below:

Harry	Hacker	*PHONE*	*TOPONYM*	Boulevard	*ZIP*	London

where labels denoting ontological categories are enclosed between stars. To this point, s_1 undergoes two levels of contextual analysis.

First-level Classifier A classifier is generally distilled from the analysis of the relationships among textual tokens, ontological categories and, also, attributes in the context of each token within the generalized sequences at hand. In particular, being s_1 composed of six tokens, a first-level classifier is learnt from the six context representations $feature_{s_1}(a) = \langle pre_{s_1}(a),\ a\ , post_{s_1}(a)\rangle$, shown in fig. 5, where a is any token of s_1.

We suppose that the resulting classifier includes the following classification rules:

if $pre_{s_1}(a) = \emptyset \quad \wedge$
$\{*TOPONYM*, *ZIP*\} \in post_{s_1}(a)$
then $\lambda(a) = NAME$

if $a = *TOPONYM*$
then $\lambda(a) = ADDRESS$

if $\{*TOPONYM*\} \in pre_{s_1}(a) \quad \wedge$
$\{*ZIP*\} \in post_{s_1}(a)$
then $\lambda(a) = ADDRESS$

if $a = *ZIP*$
then $\lambda(a) = ZIP$

if $\{*TOPONYM*, *ZIP*\} \in pre_{s_1}(a) \quad \wedge$
$post_{s_1}(a) = \emptyset$
then $\lambda(a) = CITY$

Notice that, at this stage of contextual analysis, s_1 does not include attribute labels. Hence, reconciliation takes into account relationships among ontological labels and textual tokens. These enable the reconciliation of tokens *TOPONYM*, Boulevard, London and *ZIP*, but fail in dealing with *PHONE* and Hacker. In particular, this latter token is not covered by the rule that classified Harry, since $pre_{s_1}(\text{Hacker}) = \{\text{Harry}\} \neq \emptyset$. At the end of this step of classification, sequence s_1 assumes the following form:

[NAME]	Hacker	*PHONE*	[ADDRESS]	[ADDRESS]	[CITY]	[ZIP]

where reconciliated tokens are replaced by their corresponding attribute labels, enclosed between square brackets.

Second-level classifier Contextual analysis is reiterated to reconciliate those tokens that were not associated with a schema attribute at the end of the previous step. There are only two such tokens in s_1 and, hence, a classifier is learnt from the two context representations below:

PRE	WORD	POST
[NAME]	Hacker	*PHONE* [ADDRESS] [ADDRESS] [CITY] [ZIP]
[NAME] Hacker	*PHONE*	[ADDRESS] [ADDRESS] [CITY] [ZIP]

We here assume that the resulting classifier consists only of the following rule:

if $\{NAME\} \in pre_s(a) \quad \wedge$
$\{ADDRESS, ZIP\} \in post_{s_i}(a_{ij})$
then $\lambda(a_{ij}) = NAME$

Such a rule further generalizes s_1 into

[NAME]	[NAME]	*PHONE*	[ADDRESS]	[ADDRESS]	[CITY]	[ZIP]

Notice that *PHONE* is still not reconciliated, since no the classification rule does not apply to it.

5.3 Postprocessor

The postprocessor rebuilds the original sequence s_1, by fitting its corresponding tokens in a suitable structure defined by the descriptor $R = \{NAME, ADDRESS, ZIP, CITY\}$.

NAME	ADDRESS	ZIP	CITY
Harry Hacker	Northern Boulevard	3001	London

Notice that the structure above exactly complies with R. However, in some cases, it may be useful to add an extra column *NOISE*, to the purpose of tracing all the original tokens. This would correspond to the figure below:

NAME	ADDRESS	ZIP	CITY	(NOISE)
Harry Hacker	Northern Boulevard	3001	London	3482598781

6 Experimental Evaluation

In this section we describe the experimental evaluation we performed on the proposed methodology. Experiments were mainly aimed at evaluating the classification accuracy obtained by the progressive classification methodology nested in the RecBoost approach, and to evaluate its dependency from the set of parameters which are needed to tune the system.

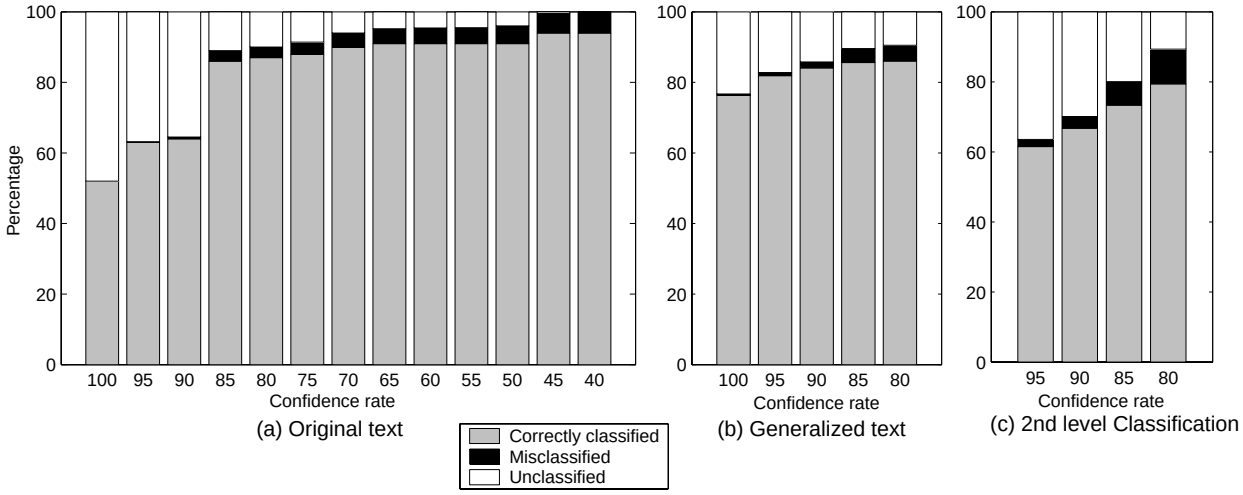


Fig. 5. Comparison of classification results

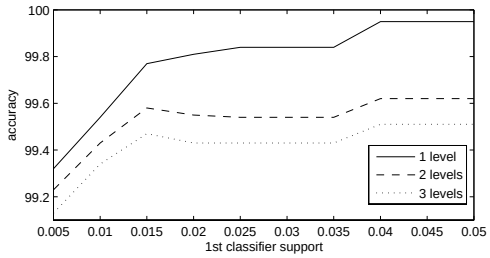
To this purpose, we tested our system on a real-life demographic database, containing information about the issue-holders of credit situations in a banking scenario. The dataset consisted of 24,000 sequences, with an average of 8 tokens per sequence. We used 4,000 sequences to exploit classification, and exploited the remaining 20,000 for testing purposes. In summary, the total number of tokens to exploit for validation was 162,879. The schema to reconcile consisted in the fields *Name*, *Address*, *Zip*, *State/Province*, and *City*.

In an initial set of experiments, we classified the data without exploiting ontologies and multiple classification stages. In these trials, the support constraint was fixed to 0.5%, with confidence ranging from 40% to 100%. Figure 5(a) describes the result of classification. Each bar in the graph describes the percentage of correctly classified tokens, together with the percentages of misclassified and unclassified tokens. As we can see, an acceptable degree of accuracy is obtained by keeping the confidence rate be under 85%. By using a confidence value of 40%, the percentage of correctly classified tokens reaches the 93% but all the remaining data are misclassified.

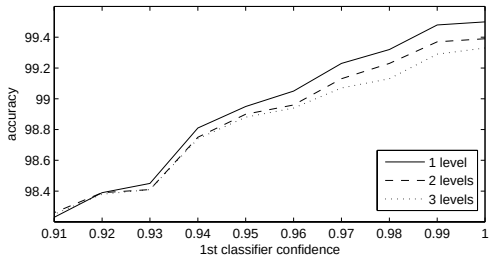
Figure 5(b) describes the accuracy of the classifier with the adoption of a domain-specific concept hierarchy. The benefits connected with the exploitation of an ontology are evident. It is worth noticing that, in both the examined cases a confidence rate of 100% guarantees a percentage of misclassified data which is nearly zero.

A further substantial increase in classification accuracy is obtained with the adoption of progressive classification. Figure 5(c) describes the result obtained by applying a second-level classifier to the unclassified cases of a previous stage of classification. In detail, the input of the second-level classifier is the output of the first-stage classifier built by using a support of 0.5% and a confidence of 100%. In both stages of classification, an ontology is used to support both preprocessor and generalizer. Again, suppport was set to 0.5% and the confidence value was ranged between 95% and 80%. As shown in figure 5(c), with a confidence threshold of 95%, the second-level classifier is able to correctly classify 62% of the data unlabeled in the previous stage, keeping the misclassification rate around 2%. By combining such a result with the results of the first-level classifier, we obtain nearly the 91% of correctly classified data, less than 1% of misclassified data and nearly the 8% of unclassified data.

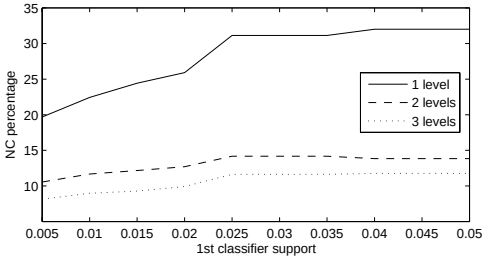
In general, a multi-stage classification leads to an increase in the number of correctly classified tokens. Notice, however that, since each stage of classification introduces a certain amount of misclassified tokens, the overall misclassification rate is increased. In order to study such an increase rate, figures 6 and 7 show both the variations in accuracy and in the number of unclassified tokens, using either one, two or three stages of classification. Here, accuracy is defined as the rate of correctly classified tokens, w.r.t. the total of classified. The graphics are plotted for different values of confidence and support of the first-level classifier. (In particular, in figures 6(b) and 7(b), the support is set to 0.5% and a varying confidence, wherease figures 6(a) and 7(a) exhibit a confidence fixed to 98% and a varying support). Support and confidence constraints for the other stages are fixed to 0.5% and 98% respectively. As we can see, the



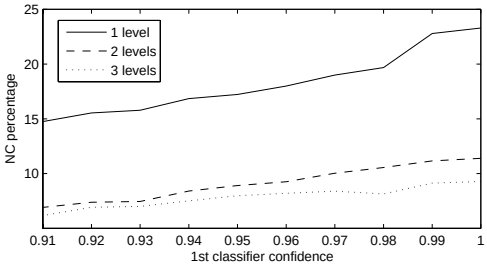
(a) Accuracy vs. support threshold



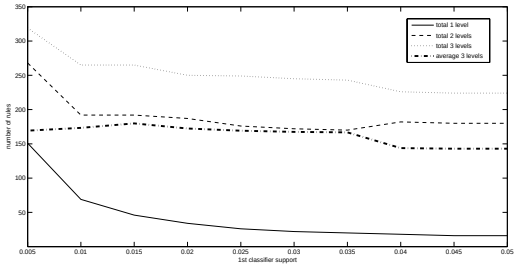
(b) Accuracy vs. confidence threshold



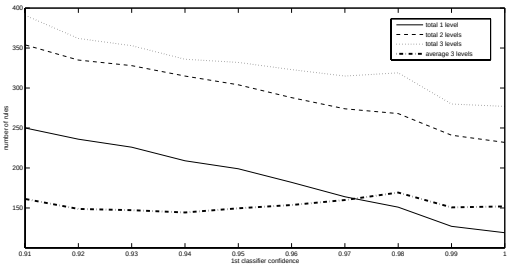
(a) Rate of unclassification vs. support threshold



(b) Rate of unclassification vs. confidence threshold



(a) Number of discovered rules vs. support threshold



(b) Number of discovered rules vs. confidence threshold

Fig. 8. Size of classifiers and average number of rules applied

loss in accuracy is limited, and the gain in terms of unclassification rate is relevant. In particular, the difference between the exploitance of one and two levels of progressive classification is quite significative.

Since the efficiency of the approach is related to the number of rules to apply, in Figure 8 we study the number of rules involved in the classification by exploiting either one, two or three stages of classification. In particular, in figure 8(a) the first level classifier confidence is kept at 98% and the support varies between 0.5% and 5%, whereas in figure 8(b) the first level classifier support is fixed at 0.5% and the confidence varies between 91% and 100%. In general, a drecrease in the support or the confidence causes an increase in the overall number of classification rules discovered. However, from experimental evaluations, it emerges that the average number of rules actually applied in the classification process does not significantly vary. This is testified by the bold hatched line in both subfigures, which represent such an average value. As we can see, the number of rules applied fluctuates around 50% of the total number of rules obtained with maximum support and confidence constraints.

Finally, table 9 shows the confusion matrix obtained by fixing support and confidence respectively to 0.05% and 98%. As we can see, errors are mainly due to the misclassifications involving the *Address* attribute. This is somehow expected, since in principle such an attribute should include the majority of

tokens in a sequence. Notwithstanding, the table exhibits significant values of precision and recall for all the involved attributes.

	Name	Address	Zip	State/Province	City	Unclassified
Name	51175	287	0	0	5	4822
Address	541	51678	0	0	217	4364
Zip	0	0	12819	0	0	0
State/Province	0	0	0	11374	0	0
City	9	242	0	0	21267	4079

Fig. 9. Confusion matrix on the overall system

7 Discussions and Future Works

The contribution of this study is RecBoost, a novel approach to schema reconciliation, that fragments free text into tuples of a relational structure with a specified attribute schema. Within RecBoost, the most salient features are the combination of ontology based generalization with rule-based classification for more accurate reconciliation, and the adoption of *progressive classification*, as a major avenue towards exhaustive text reconciliation. An intensive experimental evaluation on real-world data confirms the effectiveness of our approach.

Three main directions are worth further research. Precisely, we plan to investigate the development of an unsupervised approach to the induction of an attribute descriptor from a free text. This would still allow reconciliation, even in the absence of any actual knowledge about the textual information at hand.

Furthermore, the identification of a fully-automated technique for setting the parameters of *progressive classification*, in terms of required classification stages, is a valuable enhancement. Such a facility would indeed enable to naturally fix the parameters of the system, on the basis of the inherent features of the text at hand, rather than relying on pre-specified estimates.

Finally, we intend to examine the exploitation of RecBoost in the context of the Entity Resolution process, to the purpose of properly filling in missing fields and rectifying both erroneous data-entry and transpositions oversights.

References

1. B. Adelberg. NoDoSE: A Tool for Semi-Automatically Extracting Semistructured Data from Text Documents. In *Proc. ACM SIGMOD Conf. on Management of Data*, 1998.
2. E Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *Proc. ACM SIGKDD Conf. On Knowledge Discovery and Data Mining*, pages 20–29, 2004.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases*, pages 487–499, 1994.
4. D. M. Bikel, S. Miller, R. L. Schwartz, and R. M. Weischedel. Nymble: a high-performance learning name-finder. *CoRR*, cmp-lg/9803003, 1998.
5. V. R. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *Proc. ACM SIGMOD Conf. on Management of Data*, 2001.
6. E. Brill. A simple rule-based part of speech tagger. In *Proc. 3rd Conf. on Applied Natural Language Processing*, pages 152–155, 1992.
7. E. Brill. Transformation-based error-driven learning and natural language processing: A cased study in POS tagging. *Computational Linguistics*, 21(4):543–565, 1995.
8. M.E. Califf and R.J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proc. 16th Nat. Conf on Artificial Intelligence (AAAI'99)*, pages 328–334, 1999.
9. M. Cochinwala, S. Dalal, A.K. Elmagarmid, and V. S. Verykios. Record matching: Past, present and future, 2005.
10. S. Flesca et al. Web wrapper induction: A brief survey. *AI Communications*, 17(2), 2004.
11. L. Gu, R. A. Baxter, D. Vickers, and C. Rainsford. Record linkage: Current practice and future directions. Technical Report 03/83, CSIRO Mathematical and Information Sciences, 2003.
12. J. Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*, 6, 1992.
13. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. 4th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 80–86, 1998.

14. C.D. Manning and C. Schultze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.
15. L. Marquez, L. Padro, and H. Rodriguez. A machine learning approach to POS tagging. *Machine Learning*, 39(1):59–91, 2000.
16. A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th Int. Conf. on Machine Learning*, pages 591–598, 2000.
17. B. Merialdo. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–172, 1994.
18. S. Mukherjee and I. V. Ramakrishnan. Taming the unstructured: Creating structured content from partially labeled schematic text sequences. In *Proc. CoopIS/DOA/ODBASE Int. Conf.*, pages 909–926, 2004.
19. L. R. Rabiner. A tutorial on hidden markov models and selected application in speech recognition. *Proc. of IEEE*, 77(2):257–286, 1989.
20. S. Soderland. Learning information extraction rules for semi/structured and free text. *Machine Learning*, 34:233–272, 1999.
21. R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. 21th Int. Conf. on Very Large Databases*, pages 407–419, 1995.
22. P. Tapanainen and A. Voutilainen. Tagging accurately - don't guess if you know. In *Proc. 4th Conf. on Applied Natural Language Processing*, pages 47–52, 1994.
23. W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.