Knowl Inf Syst (2008) 15:285–320
DOI 10.1007/s10115-007-0085-3

REGULAR PAPER

Boosting text segmentation via progressive classification

Eugenio Cesario · Francesco Folino · Antonio Locane · Giuseppe Manco · Riccardo Ortale

Received: 29 November 2005 / Revised: 26 February 2007 / Accepted: 28 March 2007
Published online: 22 June 2007

Abstract A novel approach for reconciling tuples stored as free text into an existing attribute schema is proposed. The basic idea is to subject the available text to *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 an 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.

Keywords Schema reconciliation · Text segmentation · Classification

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

E. Cesario · F. Folino · A. Locane · G. Manco (✉) · R. Ortale
ICAR-CNR, Via Bucci 41c, 87036 Rende (CS), Italy
e-mail: manco@icar.cnr.it

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 [11]. 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* [7,10,23], 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, thus being specifically targeted at handling with those textual fragments not reconciled yet.

The main contribution of this paper is thus a methodological approach in which a strict cooperation between ontology-based generalization and rule-based classification is envisaged, which allows to reliably associate terms in a free text with a corresponding semantic category. A key feature is the introduction of *progressive classification*, which iteratively enriches the available ontology, thus allowing to incrementally achieve accurate schema reconciliation. This ultimately differentiates our approach from previous works in the current literature, which adopt schemes with fixed background knowledge, and hence hardly adapt to capture the multi-faceted peculiarities of the data under investigation. Moreover, due to the variable number of classification stages, RecBoost gives the user finer control over the trade-off between accuracy (i.e., the proportion of correctly classified tokens w.r.t. the classification behavior of the overall RecBoost system) and recall (i.e., the proportion of correctly classified tokens w.r.t. the actual tokens to reconcile). In practice, the user can choose a classifier with a trade-off satisfying the requirements of the specific application.

Still, the approach is further strengthened by the adoption of *local rule-based classification models*, i.e., patterns of term co-occurrence associated with specific class labels. Local models work practically well in combination with progressive classification, since they only handle the local specificities they are able to cope with, and postpone the unknown cases to subsequent classification stages. By contrast, traditional approaches from the literature exploit global classification models, which are more prone to overfitting when dealing with the several contrasting specificities occurring across individual sequences. In addition, the combination of rule-based classification models with domain-specific ontologies makes the generic RecBoost classifier very intuitive, i.e., easier to interpret than state-of-the-art probabilistic methods, such as DATAMOLD [3] and Mallet [18].

The outline of the paper is as follows. Section 2 introduces the basic notation for the problem we face; next, it continues by covering details on the process adopted to learn a

(a) Unstructured data

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

(b) Reconciled data

	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

Fig. 1 **a** Unstructured data. **b** Reconciled data

generic rule-based classifier and, then, proceeds to examine the RecBoost methodology. The architecture of RecBoost is discussed in Sect. 3. The overall RecBoost methodology is then elucidated in Section 4. Sect. 5 presents the results of an intensive experimental evaluation. Section 6 overviews and evaluates works from the literature, that are most closely related to our study. Finally, Sect. 7 draws some conclusions and highlights a number of interesting directions, that are worth further research.

2 Text segmentation with RecBoost

To formalize the Schema Reconciliation problem, we assume the basic definitions in the following section.

2.1 Notation and preliminaries

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 attribute labels. A descriptor corresponds to a database schema, with the simplification that, for each attribute label A_i, domain information is omitted. 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) $S = \{s_1, \ldots, s_m\} \subseteq \mathcal{M}^*$, we want to segment each sequence s_i into subsequences $s_i^1, \ldots, s_i^n$, 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 cities, in a proper schema with specific fields for each category, as exemplified in Fig. 1.

Text reconciliation can be profitably employed in several contexts. We briefly mention a wide range of major applications, to provide a taste of the generality of our approach. The harmonization of 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. Further

applicative scenarios requiring to deal with the schema reconciliation problem include processing bibliographic records, collections of information about products, medical sheets, and so forth.

A typical applicative setting which motivates our work is the mentioned entity resolution problem, which roughly consists in discovering/disambiguating duplicate tuples [7,10,21]. When tuples consist of free text, which however contains a hidden structure, tuple disambiguation can be accomplished by exploiting either exact matching techniques, based on specific segments of the strings, or simpler (fuzzy) techniques, that ignore segmentation. Clearly, exact matching is more reliable, provided that the original text is correctly segmented. Consider, e.g., the strings

s_1	Jeff, Lynch, Maverick, Road, 181, Woodstock
s_2	Jeff, Alf., Lynch, Maverick, Rd, Woodstock, NY

that clearly represent the same entity. A correct segmentation of the strings would eventually ease the task of recognizing the similarity.

	NAME	ADDRESS	CITY	STATE
s_1	Jeff, Lynch	Maverick, Road, 181	Woodstock	
s_2	Jeff, Alf., Lynch	Maverick, Rd	Woodstock	NY

In principle, various schemes may be followed to catch string resemblance. Without loss of generality, we here focus on two basic approaches, namely Jaccard similarity and weighted Jaccard similarity. The former measure catches resemblance between any two strings by measuring their degree of overlap, i.e., the proportion of common tokens. Formally, the Jaccard similarity between the above strings s_1 and s_2 is defined as

$$d_J(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$$

The latter measure weights segment matches by the relevance of the corresponding attributes. Let w_1, w_2, w_3, w_4 be four weights respectively denoting the relevance of segment matches in correspondence of the attributes NAME, ADDRESS, CITY and STATE. The definition of the weighted Jaccard similarity between the foresaid strings s_1 and s_2 is

$$d'_J(s_1, s_2) = \frac{\sum_{i=1}^{4} w_i d_J(a_i, b_i)}{\sum_{i=1}^{4} w_i}$$

where entities a_i and b_i denote two corresponding segments, respectively within s_1 and s_2.

It is easy to see that $d_J(s_1, s_2) = 0.44$. Such a result does not fully reflect the evident similarity of s_1 and s_2, which may generally prevent their disambiguation. To overcome such a limitation, it is sufficient to reasonably assume that a low degree of overlap between string segments corresponding to the ADDRESS and STATE attributes should not heavily penalize the overall similarity. Indeed, multiple addresses can be associated to an individual in a same city, whereas the latter resides in only one state. By accordingly fixing $w_1 = 0.5$,$w_2 = 0.1$,$w_3 = 0.35$ and $w_4 = 0.05$, it follows that $d'_J(s_1, s_2) = 0.71$, which more appropriately reflects the actual resemblance between the two strings under comparison. This confirms that exact matching techniques enable more effective string de-duplication, whenever the original strings can be accurately segmented.

Thus, a deduplication system adopting a text segmentation methodology, as described in the figure below, would effectively leverage its performance, provided that the embedded

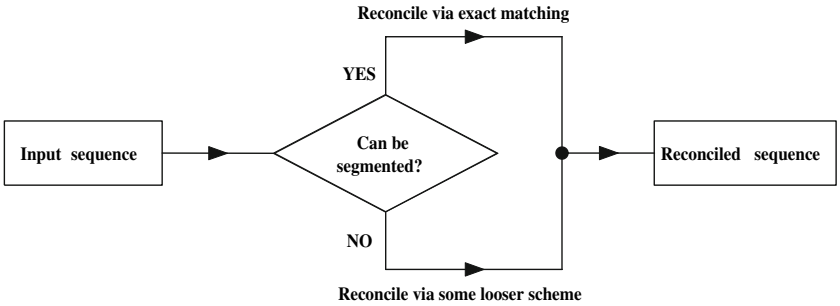
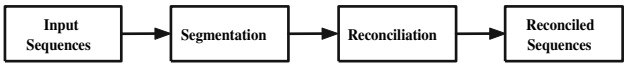


Fig. 2 Example reconciliation

segmentation methodology is reliable.



Reliability here has a strict meaning: strings should be correctly segmented, and errors in segmenting should be absolutely avoided. Indeed, a wrong segmentation would likely result in a worsening of the deduplication effectiveness, even when compared to simpler schemes. As an example, let us consider the following wrong segmentation of the above strings:

	NAME	ADDRESS	CITY	STATE
s_1	Jeff, Lynch Maverick	Road	181	Woodstock
s_2	Jeff, Alf.	Lynch Maverick, Rd	Woodstock	NY

In such a case, it still holds that $d_J(s_1, s_2) = 0.44$, whereas $d'_J(s_1, s_2) = 0.125$. In other words, though previously shown more effective, weighted Jaccard similarity now reveals unreliable. Instead, simpler schemes disregarding segmentation, such as basic Jaccard similarity, could still somehow enable string disambiguation in an acceptable way.

The point is the trade-off between precision (i.e., the capability of correctly segmenting a tuple) and recall (i.e., the capability of segmenting a whole). It is clear that classification systems exhibiting high precision, even at the cost of low recall, can be safely embedded into a deduplication system, according to the scenario described in Fig. 2.

In this scenario, a text segmentation system has two choices: either a sequence can be safely segmented, and the result of the segmentation is reliable, or the segmentation result is not affordable. In the former case, exact matching techniques based e.g., on weighted Jaccard similarity can be applied, whereas the latter case can still provide a reconciliation, which is however based on fuzzier schemes. Unfortunately, the state-of-the-art approaches from current literature, based on probabilistic modeling, do not properly fit the above scheme, since they foresee a segmentation even in presence of high uncertainty.

In this paper we discuss RecBoost, an approach for contextualized reconciliation, that moves away from probabilistic modeling. The idea is to first foresee a segmentation of textual sequences into tokens and, then, to perform a token-by-token classification, that involves the analysis of the surrounding context. This basic task is at the heart of *progressive classification*, i.e., a strategy for text reconciliation, consisting in the exploitation of multiple, consecutive stages of classification. At each intermediate stage, a classifier learns from the outcome of its

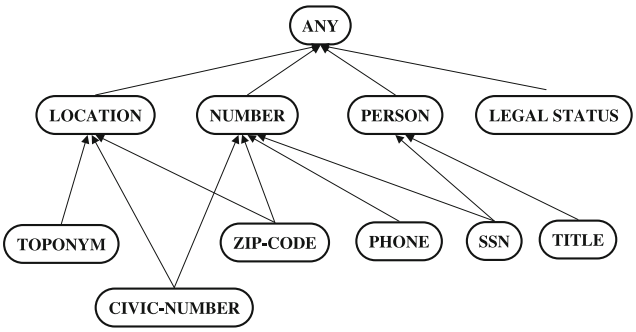


Fig. 3 A concept hierarchy for personal information in a banking scenario

antecedent how to deal with those tokens, that were not reconciled at the end of the previous stage. This ensures reconciliation effectiveness even on unknown terms.

2.2 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 labeling 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.

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 two further ontological labels, such as city and street (respectively following and preceding a), may determine the reconciliation of a with an attribute address of the schema descriptor.

2.2.1 Syntactic and ontological analysis

RecBoost exploits a user-defined domain ontology, in the style of the ones employed in [2,3], 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 whose structure is sketched below:

$$\begin{array}{ll} \textbf{if} & \mathit{Condition} \\ \textbf{then} & \mathit{Action} \end{array}$$

$\mathcal{L}$ 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 3 shows an exhaustive set of concepts and their hierarchical relationships. $\mathcal{A} = \{r_1, \ldots, r_h\}$ is a set of rules, that 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. Formally, the generic rule $r_i \in \mathcal{A}$ relates a basic textual token

with some corresponding ontological concept in $\mathcal{L}$, i.e., $r_i : \mathcal{M}^* \rightarrow \mathcal{L}$. Generally, the prior definition of a number of rules allows to properly deal with several tokens in a wide variety of applicative settings, with no substantial human effort.

As to the interpretation of ontological rules, *Condition* specifies a pattern-matching expression defined over the tokens of a sequence, and *Action* specifies some actions to take on such tokens. We here focus on two main actions, exemplified in the context of the ontological rules $\mathcal{A} = \{r_1, r_2, r_3\}$ adopted for the concept hierarchy of Fig. 3. As shown in the following illustration, rules r_1 and r_2 involve both type of actions:

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

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 token a

In general, *relabeling* actions, such as r_1, substitute a token (or a set of tokens) with a concept in $\mathcal{L}$. *Restructuring* actions, such as r_2, operate on a set of tokens, by applying basic transformation operations (such as deleting, merging or segmenting).

Rules can also exploit user-defined dictionaries. As an example, the below rule r_3 specifies that each token appearing in the set DICTIONARY of all known toponyms (which comprises, e.g., `street`, `road`, `blvd`, etc.) can be generalized by the category TOPONYM in $\mathcal{L}$.

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

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 $\mathcal{A}$. Notice that multiple matching preconditions can hold for the same set of tokens. This potential ambiguity is solved via a user-defined order over the rules in $\mathcal{A}$: when multiple rules can be applied, the first rule is chosen, and the others are ignored. In the above example, both rules r_1 and r_2 can be potentially applied to a sequence of digits. However, a token containing four digits can be interpreted as a zip code if and only if it is not followed by a new number (in which case, the former token has to be interpreted as an area code within a phone number). Thus, in order to disambiguate rule selection, r_2 is given a precedence on r_1, so that to initially favor the attempt at generalizing longer digit sequences. Second, the available tokens in each sequence are further generalized by an ad-hoc exploitation of the hierarchy described by the $\lhd$ relation. The exploitation is a direct result of a cooperation with contextual analysis, which reconciles tokens in S' as described in the next subsection.

2.2.2 Contextual analysis

This step is meant to associate tokens in S with their corresponding attributes in R. We approach the problem from a supervised learning perspective. Formally, we assume that there exists a partial 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 $a_i \in s$, we exploit information about its *context*. More specifically, the context $feature_s(a_i)$ of a generic token $a_i \in s$, is the set of all the items preceding and following a_i in s and is formally defined as follows:

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

In the above notation, $pre_s(a_i)$ is the fragment of $feature_s(a_i)$ that precedes a_i. Dually, $post_s(a_i)$ indicates the context segment that follows a_i.

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 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 $C : (\mathcal{M} \cup \mathcal{L} \cup R)^* \mapsto \mathcal{M} \mapsto R$, specified by rules such as the one sketched below:

$$\begin{array}{ll} \textbf{if} & Condition \\ \textbf{then} & \lambda_s(a) = Class \end{array}$$

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 $\mathcal{M} \cup \mathcal{L} \cup R$. The latter two conditions strictly relate token reconciliation with context inspection. Indeed, condition $v \in pre_s(a)$ (risp. $v \in post_s(a)$) narrows context analysis to what actually precedes (risp. follows) token a.

In the process of distilling a rule-based classifier from a training set T, 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 rule mining techniques naturally fit the domains under consideration.

A variant of the Apriori algorithm [22] is exploited in order 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 [15], which allows an effective heuristic search for the most accurate association rules. Succinctly, its basic idea is to sort the extracted associations by exploiting a precedence operator $\prec$. 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 (i) the confidence of r_i is greater than that of r_j, or (ii) their confidences are the same, but the support of r_i is greater than that of r_j, or (iii) both confidences and supports are the same, but r_i is shorter than r_j. Hence, a classifier can be formed by choosing a set of high precedence rules such 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 case in $\mathcal{D}$, when it is chosen.

The resulting classifier can be modeled 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.

We revised the scheme of [15] by implementing a post-processing strategy, which aims at (1) further improving the classification accuracy of the discovered rules, and at (2) reducing the complexity 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 assess 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 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.

3 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*, i.e., tokens for which no rule precondition holds. 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 a concept hierarchy mitigates such a drawback and, indeed, it has already been adopted in traditional approaches based on HMM [2,3]. 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 $\mathcal{P} = \{C_1, \ldots, C_k\}$ of rule-based classifiers is exploited to this purpose. At the generic step $i, i = 2, \ldots, k$, a classifier C_i is specifically learnt to classify all those tokens, that were not reconciled at the end of step $i - 1$. The length k of the classification pipeline is chosen so that to achieve accurate and exhaustive classification. Conceptually, this requires to minimize the overall number of both misclassified and unclassified tokens. In practice, a further classification stage is added to $\mathcal{P}$ whenever such values do not meet application-specific requirements, such as in the case where the misclassification rate is acceptable, but the unclassification rate is not satisfactory.

The generic classifier C_i can be formally described as a partial mapping $C_i : (\mathcal{M} \cup \mathcal{L} \cup R)^* \mapsto \mathcal{M} \mapsto R$, and its construction relies on a specific training set T_i, that is obtained from T_{i-1} by adding domain information provided by C_{i-1}. Given any sequence $s \in T_{i-1}$, C_i is learnt from the evaluation of the set X_s of unknown tokens, i.e., the set of those tokens in s, that are not covered by any rule of C_{i-1}. This is accomplished 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 $r \in C_{i-1}$ such as the one below:

$$\begin{array}{ll} \textbf{if} & \mathit{Condition} \\ \textbf{then} & \lambda_s(a) = \mathit{Class} \end{array}$$

is transformed into a labeling rule r', having the following structure:

$$\begin{array}{ll} \textbf{if} & \mathit{Condition} \\ \textbf{then} & \textbf{replace } a \text{ with } \mathit{Class} \end{array}$$

The new rule r' is then 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 $\langle \mathit{feature}_s(a), \lambda_s(a) \rangle$. Notice that there is a direct correspondence between the context $\mathit{feature}_s(a)$ computed at step i and the context computed at step $i - 1$. Indeed, the new context $\mathit{feature}_s(a)$ 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. 4. In the following, we explain the role played by each of the aforementioned modules.

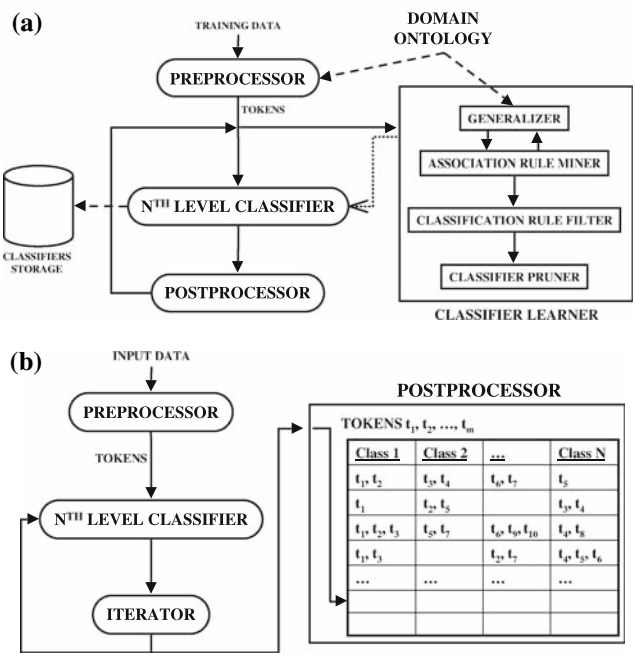


Fig. 4 Training **a** and classification **b** phases in the RecBoost methodology

3.1 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 phase 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 a whole, rather than as sequences of characters 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 $\mathcal{G}$.

3.2 Classifier learner

The *classifier learner* is responsible for producing an optimal set of classification rules, as shown in Fig. 4a. It consists of four main elements: a *generalizer*, an *association rule miner*, a filter for *classification rules* and a *classifier pruner*. In particular, 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. To accomplish this task, the generalizer employs the labeling rules in $\mathcal{A}$. Next, for each label replacing a token somewhere in a textual sequence, the related concept hierarchy is inspected and the textual sequence is extended to also include the ancestors of the specific label. The latter operation is performed

by the *association rule miner*, that extracts generalized association rules from the above extended sequences. The classification rules filtered by the *classification rules filter*, which in principle could contain several redundancies (due to the exploitation of the hierarchy in the association mining step), are further postprocessed by the *classifier pruner*. The latter attempts to reduce the overall size of the discovered rules by exploiting the aforementioned attribute and rule pruning techniques.

3.3 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. 4b. This is accomplished by interpreting each (partially) reconciled sequence as a structured tuple, and organizing the tokens that have been so far reconciled as values of 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 exploited during the training phase, as shown in Fig. 4, to yield 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}.

4 An illustrative example

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

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 containing all known toponyms, and a domain-specific ontology $\mathcal{G} = \langle \mathcal{L}, \lhd, \mathcal{A} \rangle$, where $\mathcal{L}$= { PHONE-NUMBER, SSN, TOPONYM, ZIP-CODE } and $\mathcal{A}$ 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 a 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 of 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.

4.1 Preprocessing

The input textual sequences are suitably tokenized. This is accomplished by exploiting the presence in the original text of domain-specific delimiters such as single or double quotes, hyphens, dots, brackets and blanks. After segmentation, such delimiters become spurious characters, i.e., play no further role in the reconciliation process, and are hence ignored.

The output of this step, with respect to the hypothesized data, is represented 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.

4.2 Progressive classification

Progressive classification divides into syntactic and contextual analysis.

4.2.1 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, where labels denoting ontological categories are enclosed between stars.

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

To this point, s_1 undergoes two levels of contextual analysis, where at each level a suitable set of rules is applied.

4.2.2 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, we suppose that the classifier resulting from the learning phase includes the classification rules listed below:

$$r_5: \quad \begin{array}{ll} \textbf{if} & pre_s(a) = \emptyset \quad \wedge \\ & \{*TOPONYM*, *ZIP*\} \in post_s(a) \\ \textbf{then} & \lambda(a) = NAME \end{array}$$

$$r_6: \quad \begin{array}{ll} \textbf{if} & a = *TOPONYM* \\ \textbf{then} & \lambda(a) = ADDRESS \end{array}$$

$$r_7: \quad \begin{array}{ll} \textbf{if} & \{*TOPONYM*\} \in pre_s(a) \quad \wedge \\ & \{*ZIP*\} \in post_s(a) \\ \textbf{then} & \lambda(a) = ADDRESS \end{array}$$

$$r_8: \quad \begin{array}{ll} \textbf{if} & a = *ZIP* \\ \textbf{then} & \lambda(a) = ZIP \end{array}$$

$$r_9: \quad \begin{array}{ll} \textbf{if} & \{*TOPONYM*, *ZIP*\} \in pre_s(a) \quad \wedge \\ & post_s(a) = \emptyset \\ \textbf{then} & \lambda(a) = CITY \end{array}$$

The first-level classification hence starts by classifying the tokens of s_1, according to their features. In particular, being s_1 composed of six tokens, a first-level classifier is applied against the six context representations $feature_{s_1}(a) = \langle pre_{s_1}(a),\ a\ , post_{s_1}(a)\rangle$, shown below, where a is any token of s_1.

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	-

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 the entities *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]	[ZIP]	[CITY]

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

4.2.3 Second-level classifier

To this point, as a consequence of progressive classification, the original domain information in $\mathcal{G}$ is updated to yield an enriched ontology $\mathcal{G}' = \langle \mathcal{L}, \lhd, \mathcal{A}' \rangle$. The set $\mathcal{A}'$ of ontological rules is obtained by augmenting the original one, $\mathcal{A}$, with the classification rules learnt at the end of the previous reconciliation step. Formally, $\mathcal{A}' = \mathcal{A} \cup \{r_5, r_6, r_7, r_8, r_9\}$. Recall that first-level classification rules are transformed into labeling rules before being added to $\mathcal{A}'$.

Contextual analysis is then reiterated to reconciliate those tokens that were not associated with a schema attribute at the end of the previous step. Again, we assume that a second-level classifier is learnt from the training data and, and it is composed by the following individual rule:

$$r_{10}: \quad \begin{array}{ll} \textbf{if} & \{NAME\} \in pre_s(a) \quad \wedge \\ & \{ADDRESS, ZIP\} \in post_s(a) \\ \textbf{then} & \lambda(a) = NAME \end{array}$$

There are only two tokens in s_1 that were not associated with a schema attribute and, hence, the above classifier is applied against two context representations:

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

As a result, the classifier further generalizes s_1 into the following sequence:

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

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

4.3 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 following tuple:

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

5 Experimental evaluation

In this section, we describe the experimental evaluation we performed on the proposed methodology. Experiments were mainly aimed at evaluating the effectiveness of the proposed methodology in segmenting strings. To this purpose, we accomplish the following tasks:

1. We evaluate the effectiveness of the basic rule-based classifier systems proposed in Sect. 2.2.2. Since the classification methodology represents the basic infrastructure upon which the RecBoost system bases, it is important to assess its effectiveness in the domain at hand. In particular, we evaluate two main aspects: (i) its dependency from the parameters which are needed to tune the system, and (ii) the effectiveness of the pruning strategy introduced.
2. Next, we evaluate classification accuracy obtained by the progressive classification methodology nested in the RecBoost approach, as described in Sect. 3. Our aim here is to investigate in which respect the envisaged pipeline boosts the performance of a basic classifier. We also compare our results with other state-of-the art text segmentation systems.

5.1 Experimental setup

In order to accomplish the above tasks, we considered the following datasets:

- `Addresses`, a real-life demographic database consisting of information about the issue-holders of credit situations in a banking scenario. Such a dataset is of particular interest, since it contains several fragments of noisy data. The dataset consisted of 24,000 sequences, with an average of eight tokens per sequence. The schema to reconcile consisted of the fields *Name*, *Address*, *Zip*, *State/Province*, and *City*.
- `BigBook`, a publicy-available dataset[1] consisting of a set of business addresses. Each business description consists of the six items *Name*, *Address*, *City*, *State*, *AreaCode*, and *Phone*. The dataset consists of 4,224 sequences, with ten tokens per sequence in the average. The dataset is of particular interest, since the relatively small size of the available dataset allows us to evaluate whether RecBoost is sensitive to the number of training tuples.
- `dblp`, a collection of articles extracted from the DBLP website[2]. Each entry refers to an article appeared in a Computer Science Journal, and contains information about *author*, *title*, *journal*, *volume*, *year*. We extracted 19,401 sequences, with an average sequence length of 20 tokens.

The evaluation of Recboost effectiveness requires the design of a domain-specific ontology for each of the aforementioned datasets. Specifically, the concept hierarchy devised for the `Addresses` dataset is shown in Fig. 3. This consists of 11 concepts for token generalization, suitably organized into a compact hierarchical structure. The ontological rules include rules r_1, r_2, r_3 and r_4 at Sect. 4 (as relabelling rules) and rule r_2 at Sect. 2.2.1 (as a restructuring rule).

The ontology employed for the `BigBook` dataset, shown in Fig. 5, embraces nine concepts.

[1] http://www.isi.edu/info-agents/RISE/repository.html.

[2] http://www.informatik.uni-trier.de/~ley/db/.

In such a context, no use is made of restructuring actions, so that background knowledge reduces to the relabeling rules shown next:

r_1: **if** $a \in$ DICTIONARY
then **replace** a with TOPONYM

r_2: **if** a is a token of type *ddd-dddd*
then **replace** a with PHONE NUMBER

r_3: **if** a is a token of type *(ddd)*
then **replace** a with AREA-CODE

r_4: **if** a is a digit-sequence followed by *TH* or *ST* or *ND* or *RD*
then **replace** a with STREET NUMBER

r_5: **if** a is a digit-sequence
then **replace** a with NUMBER

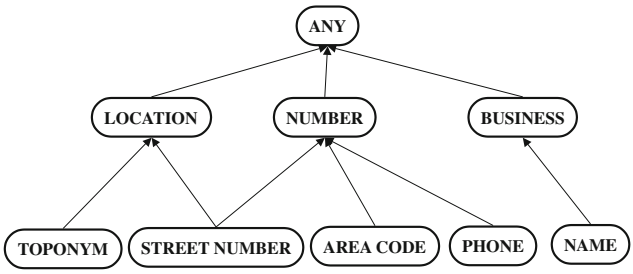


Fig. 5 The concept hierarchy for the `BigBook` dataset

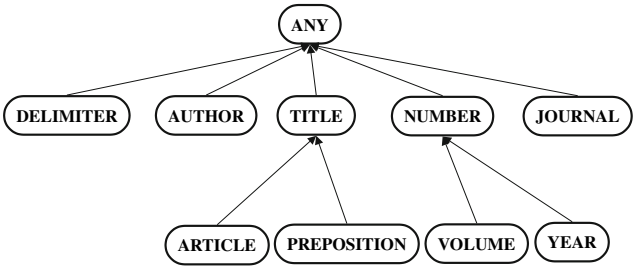


Fig. 6 The concept hierarchy for the `dblp` dataset

Finally, the ontology for the dblp dataset is shown in Fig. 6.

Again, background knowledge only involves relabeling rules, that are reported below:

r_1: **if** $a \in$ JOURNAL DICTIONARY
then **replace** a with JOURNAL

r_2: **if** a is a four-digits token
and $a \in [1950 \div 2006]$
then **replace** a with YEAR

r_3: **if** a is a digit-sequence
then **replace** a with NUMBER

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

r_5: **if** $a \in$ GRAMMATICAL-ARTICLE DICTIONARY
then **replace** a with ARTICLE

r_6: **if** $a \in$ PREPOSITION DICTIONARY
then **replace** a with PREPOSITION

Notice that the definition of the above rules relies on a number of domain-specific dictionaries. In particular, JOURNAL DICTIONARY includes several alternative ways of denoting a journal article, such as *j.*, *journal*, *trans.* and *transaction*. GRAMMATICAL- ARTICLE DICTIONARY groups English-language articles *a*, *an* and *the*. Similarly, PREPOSITION DICTIONARY collects commonly used prepositions, such as *by*, *to*, *with* and *via*. DELIMITER DICTIONARY is a set of token delimiters, that comprises ', ", :, ,, -, ., and *.

It is worth noticing that the analysis of the above domain-specific ontologies reveals a key feature of RecBoost methodology. Roughly speaking, it can be easily employed for pursuing text reconciliation in a wide variety of applicative settings, by simply providing a domain-specific concept hierarchy along with a corresponding compact set of ontological rules. The overall process of ontology design is rather intuitive and does not require substantial effort by the end user.

The evaluation of the results relies on the following standard measures which are customized to our scenario. Given a set N of tokens to classify, we define:

- the number of tokens, which were classified correctly, *TP*;
- the number of tokens, which were misclassified, *FP*;
- the number of tokens, which were not classified, *FN* — notice that this is a different meaning with respect to the standard literature.

In the following, we shall report and illustrate the above measures over the mentioned datasets. Two further important measures, however, can give an immediate and summarizing perception of the capabilities of our classification system. In particular, *Precision* (or *Accuracy*) can be defined as the number of correctly classified tokens, w.r.t. the classification behavior of the system:

$$P = \frac{TP}{TP + FP}$$

Analogously, *Recall* can be defined as the number of correctly classified tokens, w.r.t. the tokens to classify:

$$R = \frac{TP}{TP + FN}$$

Intuitively, Recall describes the locality issues, that affect the system: if a classifier contains rules which can cover all the examples, then it has 100% recall (i.e., no locality effect). Precision, by the converse, describes the accuracy of the rules contained: the higher is the error rate of a rule, the lower is its precision.

A measure which summarizes both precision and recall is the $\mathcal{F}$ measure, defined as

$$\mathcal{F} = \frac{(\beta^2 + 1)PR}{P + \beta^2 R}$$

The $\mathcal{F}$ measure represents the harmonic mean between Precision and Recall. The β term in the formula assigns different weights to the components: when $\beta = 1$ both the components have the same importance. The tuning of the β parameter is application-dependent. Here, we are interested in the cases where $\beta > 1$ (which assigns higher importance to Precision than to Recall). This is a crucial requirement of many application domains, such as the one described in Sect. 2. Hence, in the following we shall study the situations where $\beta > 1$, and in particular we are interested in the cases where β ranges into the interval $(1, 10]$.

5.2 Evaluating the basic classifier system

In an initial set of experiments, we classified the data without exploiting ontologies and multiple classification stages. In these trials, support was fixed to 0.5%, with ranging values of confidence. Figure 7 shows the outcome of classification for the three datasets. 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, the effectiveness of the classifiers strongly relies on the confidence value. In particular, low confidence values (up to 40% in both `Addresses` and `dblp`, and 60% in `BigBook`) to classify all the tokens, but the percentage of misclassified is considerably high. This is somehow expected, since low confidence values induce rules exhibiting a weak correlation between the antecedent and the consequent.

By contrast, higher confidence levels lower the misclassification rate, but the degree of unclassified tokens raises considerably. It is worth noticing that, in all the examined cases a confidence rate of 100% guarantees a percentage of misclassified data which is nearly zero. This is the *locality effect*: high confidence values produce extremely accurate rules that, as a side effect, apply only to a limited number of tokens. By lowering the confidence, we relax the locality effect (the resulting rules apply to a larger number of tokens), but the resulting rules are less accurate.

The `dblp` dataset is particularly interesting to investigate in this context, since it exhibits the worst performances. The best we can obtain in this dataset is with confidence set to 40%, which guarantees a significantly high percentage (30.52%) of misclassified tokens. A "safer" confidence value leverages the number of unclassified tokens considerably.

Figure 8 describes the accuracy of the classifier with the adoption of domain-specific concept hierarchies. We exploited the hierarchies described in Figs. 3, 5 and 6, respectively. The benefits connected with the exploitation of such simple ontologies are evident: the generalization capabilities of the classification rules are higher, thus lowering the number of unclassified tokens. Notice how the `dblp` dataset still exhibits unacceptable performances.

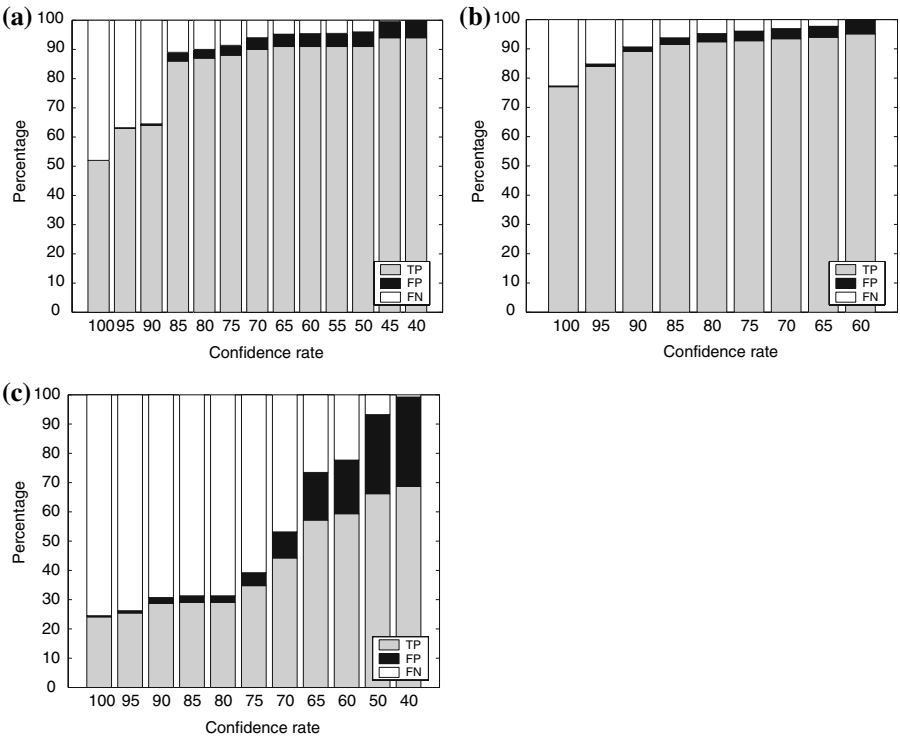


Fig. 7 Classification results, single stage of classification **a** Addresses **b** BigBook **c** dblp

Table 1 Pruning effectiveness

Confidence			100	90	80	70	60	50	40
FP	Addresses	Unpruned	0.11%	1.73%	3.93%	5.52%	6.45%	7.69%	8.05%
		Pruned	0.09%	1.53%	3.77%	5.32%	5.94%	6.47%	6.24%
	BigBook	Unpruned	0.25%	1.47%	1.50%	1.53%	1.94%	1.94%	1.95%
		Pruned	0.21%	0.70%	0.70%	0.72%	0.98%	0.98%	0.98%
	Dblp	Unpruned	0.01%	3.30%	3.81%	7.13%	14.55%	17.02%	17.12%
		Pruned	0.01%	3.26%	3.77%	7.09%	14.38%	16.06%	16.20%

Results in the above figures were obtained by exploiting the pruning steps detailed in Sect. 2.2.2. Indeed, the contribution of the classifier pruner to the misclassification rate is investigated in Table 1, which describes how the error rate changes if pruning is not applied. The effectiveness of the classifier pruner can be appreciated at lower confidence values: there, the classifier produces weaker rules, which clearly benefit of a re-examination.

5.3 Evaluating multiple classification stages

The above analysis allows us to test the effectiveness of the *progressive classification* methodology. We recall the underlying philosophy: starting from the following observations,

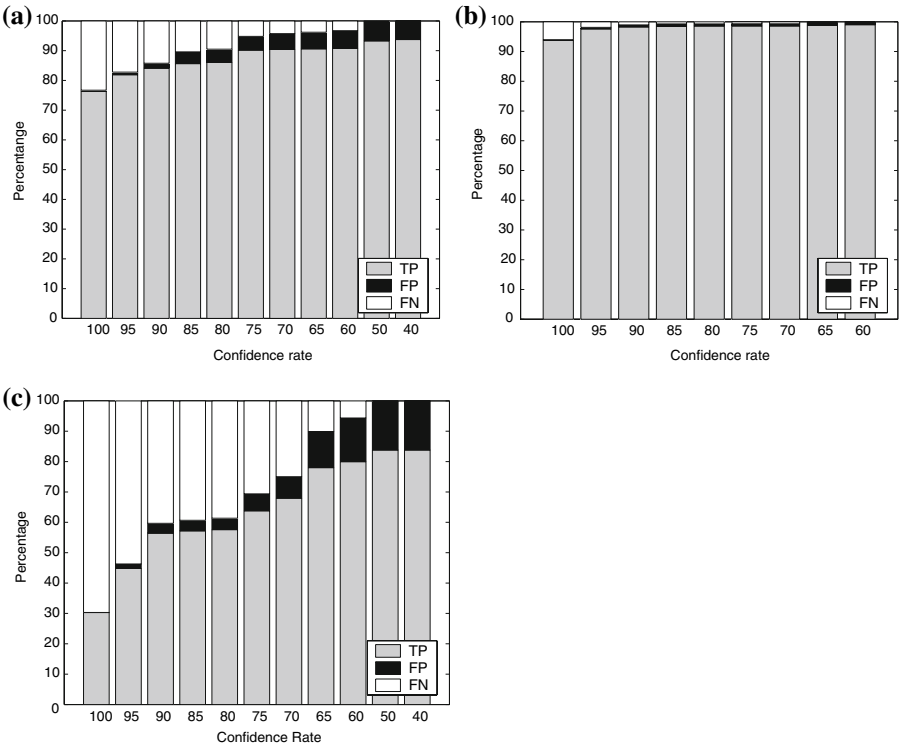


Fig. 8 Classification results with the exploitation of concept hierarchy. **a** Addresses **b** BigBook **c** dblp

– ontological analysis eases the classification task (as testified by the comparison between graphs in Figs. 7 and 8);
– a richer set of relabeling rules should in principle boost the results of classification;

the adoption of multiple classification stages, where at each stage the relabeling rules of the previous stage are enriched by exploiting the results of classification at earlier stages, should boost the performance of the overall classification process.

And indeed, Fig. 9 describes the results obtained by applying a second-level classifier to the unclassified cases of the first stage of classification. In detail, the input to the second-level classifier is the output of the first-level classifier, built by fixing support to 0.5% and a confidence to 100% (described by the first bar of each graph in Fig. 8). Again, support was set to 0.5% and confidence was ranged between 100 and 80%.

As shown in the figure, the second-level classifier is in general able to correctly classify a portion of the data, that were unlabeled at the end of the previous stage. For example, in the `Addresses` dataset, a 95% threshold allows to classify a further 62% of the (originally unclassified) data. By combining such a result with the outcome of the first-level classifier, we obtain nearly 91% of correctly classified data, less than 1% of misclassified data and nearly 8% of unclassified data. Table 2 summarizes the the cumulative results achieved by two levels of classification over the employed datasets.

The effectiveness of the second stage of classification is even more evident in the graphs of Fig. 10. The graphs depict the trend of $\mathcal{F}$ for different values of β. The graphs compare a selection of 2-level classifiers with the single-level classifier (among those shown in Fig. 8)

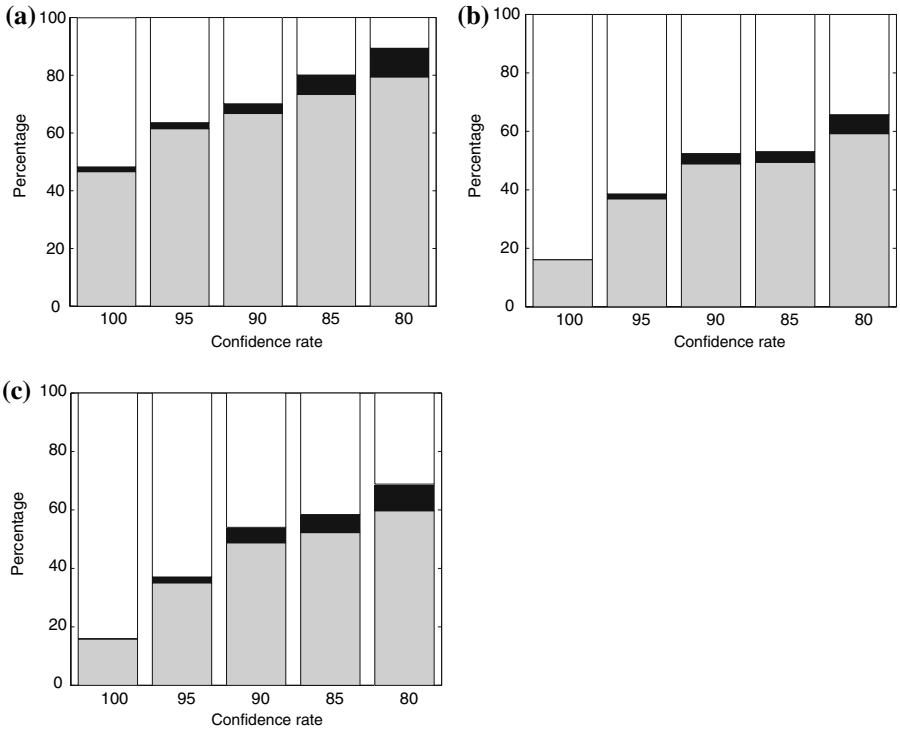


Fig. 9 Classification results at the second stage of classification **a** Addresses **b** BigBook **c** dblp

Table 2 Precision and recall at varying degrees of confidence over the selected datasets

Confidence	Addresses		BigBook		dblp	
	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
100	99.13	87.87	99.78	94.95	99.73	41.40
95	99.06	91.44	99.68	96.30	97.51	55.52
90	98.74	92.96	99.57	97.13	94.59	66.75
85	97.95	95.26	99.56	97.17	93.92	69.72
80	97.24	97.45	99.39	97.93	91.89	76.80

exhibiting the best performance in terms of *TP*. In all the cases shown, the 2-level classifiers exhibit better performances for $\beta > 2$.

Since each classification level boosts the performance of the system, two important questions raise, that are worth further investigation in the following:

1. how many levels allow to achieve an adequate performance?
2. how should the parameters at each level be tuned?

The `dblp` dataset is particularly interesting in this context, since the accuracy of RecBoost is still low after two classification levels. We start our study by investigating the number of needed classifiers. Figure 11a, b describe an experiment performed by allowing a hypothetical infinite number of levels, where at each level support was set to 1% and confidence

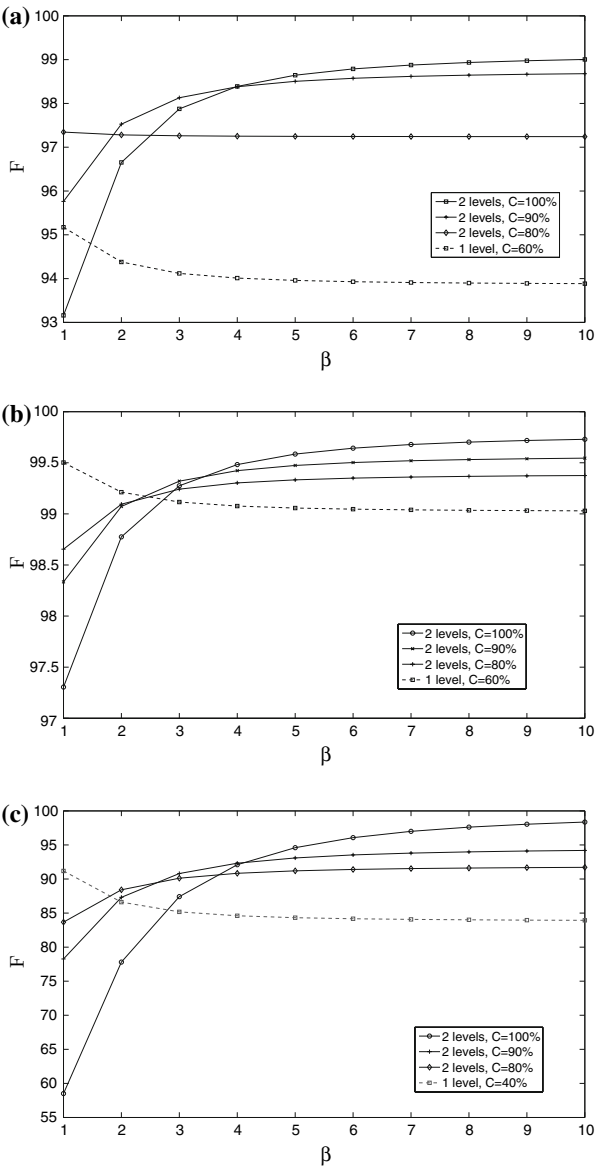


Fig. 10 Trends of $\mathcal{F}$-measures compared. **a** Addresses **b** BigBook **c** dblp

to 100%. Roughly, the strategy implemented is the following: since high confidence values bound the number of misclassified tokens, and further levels allow to recover unclassified tokens, just allow any number of levels, until the number of unclassified tokens is nearly 0.

As we can see from Fig. 11b, however, this strategy does not necessarily work: although the number of misclassified tokens is kept low, the capability of each classifier to recover tokens unclassified in the previous stages decreases. The 5th level looses the capability to further classify tokens, thus ending de-facto the classification procedure. Figure 11a shows the cumulative results at each level.

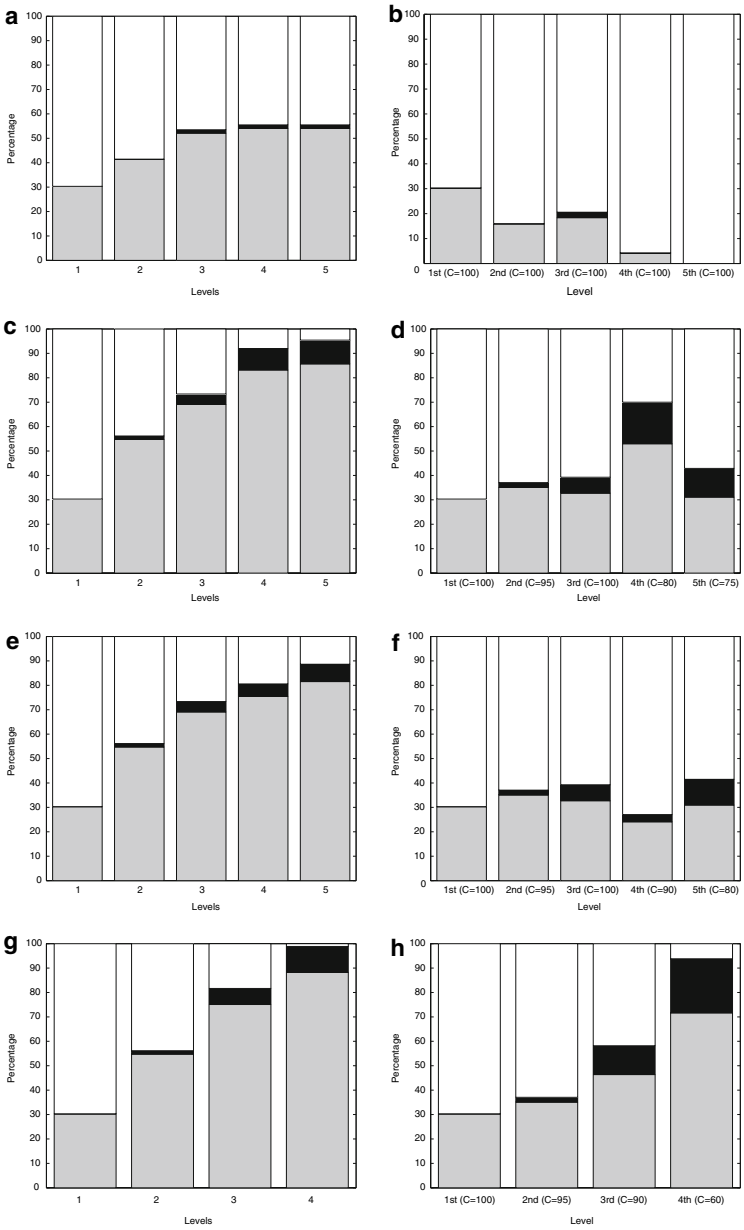


Fig. 11 Effects of multiple classification levels on `dblp`. **a** Cumulative performance, 5 levels, **b** 5 levels, Performance of single stages. **c** Cumulative performance, 5 levels, **d** 5 levels, Performance of single stages. **e** Cumulative performance, 5 levels, **f** 5 levels, Performance of single stages. **g** Cumulative performance, 4 levels, **h** 4 levels, Performance of single stages

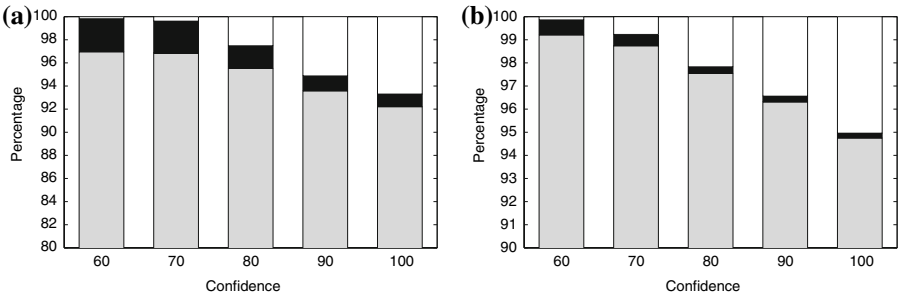


Fig. 12 Effects of multiple classification levels. **a** Addresses, **b** BigBook

Thus, an upper bound in the number of stages can be set by the classification capability of the stages themselves. A smarter tuning of the parameters which rule the performance of each single stage, allows to achieve best classification accuracy. Figure 11c, d report a different classifier, generated by fixing the following constraints: each classification stage should classify at least 30% of the available tokens, and should misclassify at most 10% (if possible). The methodology adopted for achieving this was to perform several tuning trials at each stage, by starting from the value 100% of confidence and progressively lowering it until the criterion is met. Figure 11d describes the tuning occurred at each classification stage. The constraint over the classification percentage clearly boosts the performance of each single classification stage: as a result, the overall number of classified tokens is 86.7%, with a misclassification rate of 10.2 and 3.1% unclassified tokens.

Notice that further effective strategies can be employed, by fixing e.g., different constraints: in Fig. 11e, for example, each classification stage should classify at least 20% of the available tokens, and should misclassify at most 5% of them. Figure 11g, reports a different experiment, where the number of stages is fixed to 4: here, confidence is progressively lowered, and the last stage is tuned to minimize the number of unclassified. Again, Fig. 11h describes the tuning occurred at each stage.

Similar conclusions can be drawn with the other datasets: Fig. 12, e.g., describes the results on both `BigBook` and `Addresses`. In particular, we adopted three levels (with confidence fixed to 100% in the first two levels) for `BigBook` and four levels (with thresholds 100, 100, 85% in the first three levels) for `Addresses`. The bars report the cumulative classification results when different confidence levels are applied in the last classification level.

The adoption of multiple classification stages over `BigBook` deserves further discussion about the relation between the size of the labeled data and the number of classification levels which can be defined. Each classification level should build on a separate training set (preprocessed by the preceding levels). Clearly, given a dataset D, the amount of unclassified tokens of D diminishes at subsequent levels. Hence, the size of the training set T_i required for learning rules at level i should be large enough to guarantee that an adequate number of unclassified tokens are available at that level.

Thus, the size of the training has an influence over the number of classification levels which can be defined: the larger the training set, the higher the number of significant levels. In other words, a small dataset saturates the potential of progressive classification within few levels, and adding further levels does not yield any improvements. This is what happens in the case of the `BigBook` dataset. As already mentioned, the available training set here is quite small. Thus, a classifier exhibiting 100% confidence in the last level, would produce at

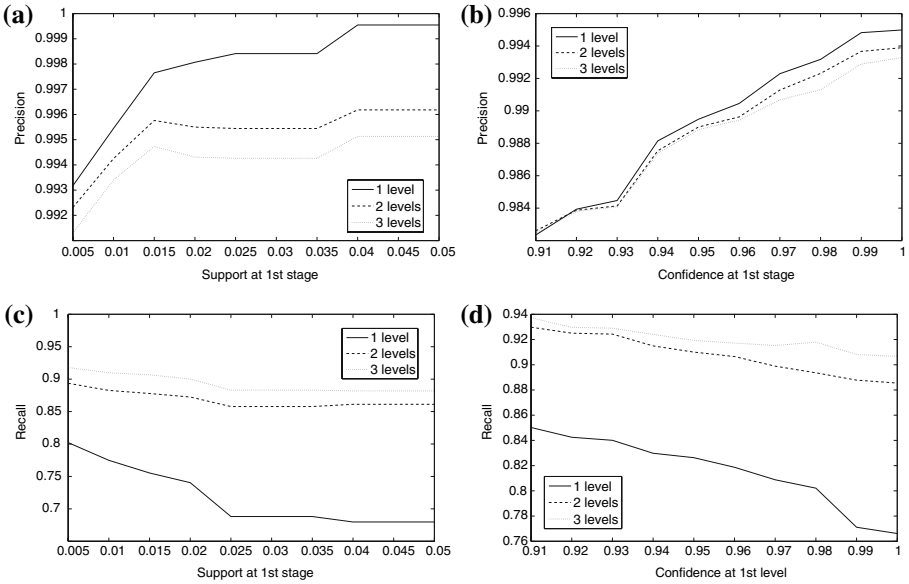


Fig. 13 Stability of Precision and Recall with variable parameter values **a** Precision vs. support threshold. **b** Precision vs. confidence threshold. **c** Recall vs. support threshold. **d** Recall vs. confidence threshold

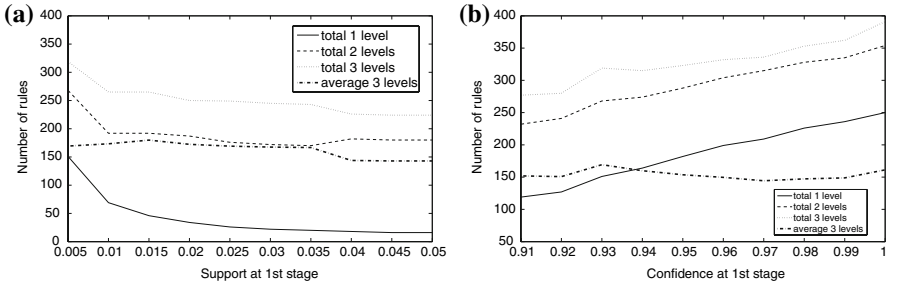


Fig. 14 Size of classifiers and average number of rules applied. **a** Number of rules vs. support. **b** Number of rules vs. confidence

most 2,500 unclassified tokens. This amount would not allow to learn a further meaningful set of rules, since such tokens distribute over different sequences and different attributes.

The conclusion we can draw is that the adoption of multi-stage classification allows to increase recall, by contemporarily controlling the decrease in the overall classification accuracy. The above figures show how a proper manipulation of the confidence threshold value over each classification stage allows to achieve this. The contribution of the support threshold is less restrictive for two main reasons: first, it should anyway be kept at very low levels, in order to enable a significant amount of rules; second, small variations are of little significance, and at most at the first level. We here provide details on a set of tests performed on a pipeline of three classifiers, over the `Addresses` dataset. In particular, for brevity sake, we investigate the effects of varying support and confidence for the first-level classifier, whereas the remaining two stages have instead both parameters fixed to respectively 0.5 and 98%.

Specifically, in Fig. 13a, c confidence is fixed to 98% and support varies, whereas in Fig. 13b, d support is set to 0.5% and confidence varies. Figure 13a, c shows that classification accuracy and recall do not significantly change, especially at higher levels. By the converse, even small variations in the confidence cause significant changes, as testified by Fig. 13b, d.

It is interesting to see that, in the above described experiments, the average number of rules which are exploited is nearly stable even on different values of support and confidence (which instead affect the number of discovered rules). Fig. 14a,b depict such a situation. In general, a decrease in support or confidence causes an increase in the overall number of discovered classification rules. 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 represents such an average value. As we can see, the number of rules applied fluctuates around 50% of the total number of rules obtained in correspondence of the maximum values of support and confidence.

5.4 Comparative analysis

The exploitation of the "recursive boosting" strategy proposed in this paper is quite new, as it relies on the capability of recovering unclassified tokens in the next stages. To this purpose, the former experiments aimed essentially at checking whether this strategy is effective. In order to asses the practical effectiveness of RecBoost, we here compare the behavior of the RecBoost methodology with consolidated approaches from the literature. To this end we preliminarily observe that, although many results are available in the literature, a direct comparison is often difficult, as different data collections and/or different ways of tuning the algorithm parameters have been used. For example, although bibliographic citations extracted from the DBLP database have been extensively used in the literature, the datasets used for the analysis were not made publicly available.

In the following we provide a comparison by exploiting the datasets described in the previous sections. We compare our system with the Mallet system [18], which provides the implementation of Conditional Random Fields [14] and with the DataMold system [3]. We refer the reader to Sect. 6 for a detailed description of the techniques underlying such systems. Both Mallet and DataMold are equipped with the same ontology and preprocessing used in RecBoost. In addition, contextual information in the CRF implemented by Mallet was provided by resorting to the Pre/Post information.

An overall comparison is shown in the graphs of Fig. 15, which plot the $\mathcal{F}$ values obtained by Mallet, DataMold, and several different instantiations of the RecBoost system. In particular, we consider the classifiers of Figs. 11 and 12, and choose, for each dataset, the three instantiations which guarantee the lowest (constrained) value of *FN*, the lowest (constrained) value of *FP*, and a "middle" value. The constraint refers to the possibility of maintaining an acceptable value of *TP*. For the `dblp` dataset, we also show an instantiation

As we can see from the figure, the gain in the $\mathcal{F}$ value is evident for $\beta > 2$. Table 3 details the results. Here we compare with Mallet, DataMold and the version of RecBoost (RecBoost[1] in the tables), relative to a single stage of classification which achieves the highest value of *TP* in Fig. 8. Mallet (and in some cases even Datamold) typically achieves a high rate of correctly classified tokens at the expense of a higher misclassification rate. Also, RecBoost[1] may achieve a higher *TP* than the approaches with multiple classification stages. However, the latter exhibit a higher affordability (which is even higher than that of Mallet and DataMold). In practice, the adoption of multiple stages allows to achieve a higher precision, at the expense of a lower recall. Clearly, a proper tuning at the higher levels makes the Rec-

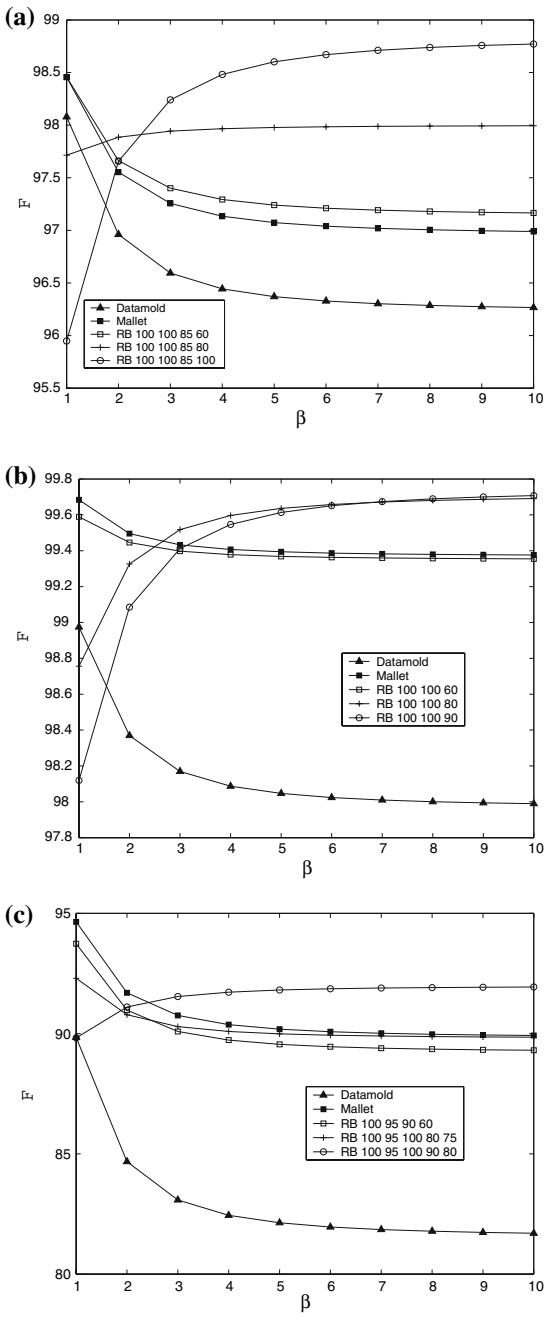


Fig. 15 Trends of $\mathcal{F}$-measures compared. **a** Addresses **b** BigBook **c** dblp

Boost system highly competitive: in `Addresses`, for example, the performance of the more conservative classifier (the one which tries to minimize *FN*) is even better than Mallet.

In practice, the recursive boosting offered by progressive classification allows to maintain a higher control over the overall misclassification rate, by forcing stronger rules which, as

Table 3 Comparison against Mallet and DataMold

Methods	Addresses							
	TP(%)	FP(%)	FN(%)	P(%)	R(%)	F_1(%)	F_2(%)	F_3(%)
DataMold	96.23	3.77	0	96.23	100	98.08	96.96	96.59
Mallet	96.96	3.04	0	96.96	100	98.45	97.55	97.25
RecBoost^1	93.74	6.24	0.02	93.76	99.98	96.77	94.94	94.34
RecBoost$^\square$	96.96	2.86	0.18	97.14	99.81	98.45	97.66	97.40
RecBoost$^+$	95.53	1.95	2.52	98.00	97.43	97.71	97.88	97.94
RecBoost$^\circ$	92.21	1.09	6.70	98.83	93.23	95.95	97.65	98.24
Methods	BigBook							
	TP(%)	FP(%)	FN(%)	P(%)	R(%)	F_1(%)	F_2(%)	F_3(%)
DataMold	97.97	2.03	0	97.97	100	98.97	98.36	98.16
Mallet	99.37	0.63	0	99.37	100	99.68	99.49	99.43
RecBoost^1	99.01	0.98	0.01	99.02	99.99	99.50	99.21	99.11
RecBoost$^\square$	99.21	0.65	0.14	99.35	99.83	99.59	99.44	99.40
RecBoost$^+$	97.55	0.28	2.17	99.71	97.82	98.75	99.32	99.51
RecBoost$^\circ$	96.31	0.25	3.44	99.74	96.55	98.11	99.08	99.41
Methods	dblp							
	TP(%)	FP(%)	FN(%)	P(%)	R(%)	F_1(%)	F_2(%)	F_3(%)
DataMold	81.55	18.45	0	81.55	100	89.83	84.67	83.08
Mallet	89.83	10.17	0	89.83	100	94.64	91.69	90.75
RecBoost^1	83.80	16.20	0	83.80	100	91.18	86.60	85.18
RecBoost$^\square$	88.20	10.66	1.14	89.22	98.73	93.73	90.97	90.09
RecBoost$^+$	85.69	9.74	4.57	89.80	94.94	92.30	90.78	90.29
RecBoost$^\circ$	81.53	7.10	11.37	91.98	87.76	89.82	91.10	91.54

a side effect, exhibit a higher *locality*. Thus, RecBoost is more reliable in scenarios where misclassifying is worst than avoiding to classify.

Finally, two major arguments emerge in favor of Recboost as a further result of our comparative analysis.

- Due to the variable number of classification stages, RecBoost gives the user better control over the trade-off between accuracy and recall. In practice, the user can choose a classifier with a trade-off satisfying the requirements of the specific application.
- The generic RecBoost classifier is easier to interpret than existing methods such as DATAMOLD [3] and Mallet [18], since it produces symbolic rules using vocabulary from a domain-specific ontology.

6 Related work

Text reconciliation 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 and rule-based techniques [4,13,16,17] have been proposed in the literature. In practice, the basic idea behind POS tagging consists in disambiguating phrases by exploiting dictionaries and analyzing the textual context surrounding each 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 [8]. 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, since 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 [1,5,21]. For example, WHISK [21] can deal with missing values and permutations of fields, but it requires a "complete" training set, i.e., a set of examples including all the possible occurrences of values.

ILP provides a solid framework for designing rule-based systems aimed at text categorization and information extraction [6,9]. In particular, a divide-and-conquer approach to the problem of learning rules for accomplishing both these latter tasks is proposed in [12]. In principle, such a technique can be employed for text reconciliation, since it allows the extraction of focussed textual fragments and their subsequent labeling. However, its exploitation for practical applications is problematic, due to the fact that rules have to explicitly locate fragment boundaries. This imposes a non-trivial learning phase, that requires two distinct sets of training examples, respectively necessary for denoting what specific (aggregates of) words can be taken into account as possible boundaries within the underlying textual data and for specifying how to label their intermediate fragments. Also, fragment extraction relies on tests on the occurrences of domain-specific words, that are either learnt from the training examples, or exhibit some degree of positive correlation to such examples. However, locating all relevant fragments determined by meaningful combinations of these words is computationally unfeasible. This imposes the exploitation of various indexing structures to accelerate the evaluation of rule predicates on the underlying text. By contrast, RecBoost pursues token-by-token reconciliation, thus overcoming all of the above issues related to fragment boundaries.

Several recent approaches to schema reconciliation rely on Hidden Markov Models (HMM) [2,3,14,19,20]. Schema reconciliation with HMM can be accomplished by learning the structure of a HMM, and applying the discovered structure to unknown examples. As an example, DATAMOLD [3] employs a training phase to learn a HMM, that consists of a set of states and directed 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. Every state of the HMM, except from the *initial* and *final* ones, represents a class label and is associated with a dictionary, grouping all the terms in the training set that belong to the class. Edges among states are associated with transition probabilities. A textual sequence can be classified if its constituting terms can be associated to states of the HMM, that form a path between the initial and final states. Precisely, DATAMOLD pursuits classification by

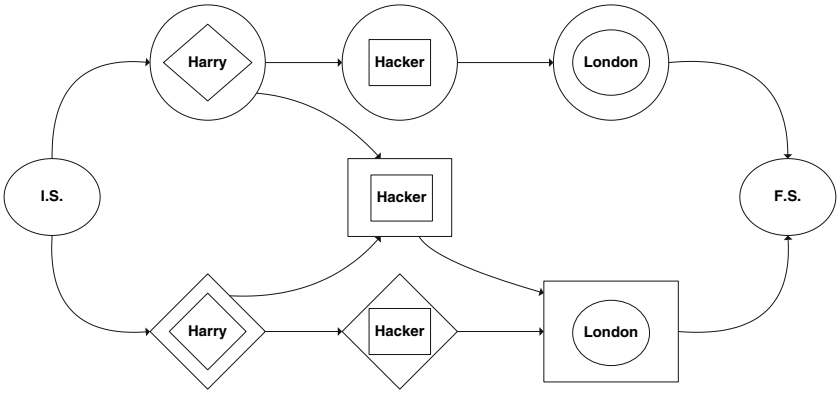


Fig. 16 Unreliable token reconciliation due to HMM topology

associating a single term to all those states, whose corresponding dictionaries include the term. Hence, a sequence of textual terms is mapped to multiple paths throughout the HMM. Transition probabilities are then exploited to identify the most probable path and, hence, to accordingly classify the terms in the sequence at hand. Clearly, those sequence, whose tokens do not form any path between the initial and final states, cannot be classified.

The effectiveness of the approaches based on HMMs strongly depends on the number of distinct terms occurring in the training set. Indeed, in order to associate terms that do not appear in the training set with a corresponding state, DATAMOLD relies on *smoothing*, i.e., on the exploitation of ad hoc probabilistic strategies.

Furthermore, the classification of individual term sequences in one step, i.e., subjected to the existence of corresponding paths throughout the automaton, is a major limitation of HMMs. Indeed, depending on the outcome of the training phase, these cannot undertake the reconciliation process, whenever a path for the sequence at hand does not exist. Also, the existence of one or more paths for a given input sequence may not determine a proper reconciliation. This latter aspect is clarified in Fig. 16, where it is shown that the HMM topology prevents the correct reconciliation of the input sequence Harry Hacker London. Notice that node labels indicate input tokens for the automaton. Moreover, the internal shape surrounding node labels corresponds to actual token classes (i.e., name/rhombus, surname/rectangle and city/circle), whereas the external shape of a node denotes the class assigned by that node to its label. Clearly, discrepancies between internal and external shapes represent reconciliation errors. The illustration shows that four paths exist in the automaton and, hence, as many alternatives to reconcile the input sequence. However, all such paths lead to erroneous reconciliations. The only sequence of states in the automaton, that would lead to a correct reconciliation, does not form a path between the ending states $I.S.$ and $F.S.$, thus being unemployable to reconciliation purposes.

Worst, HMMs represent "global classification models", since they tend to classify each term of the sequence under consideration, and hence are quite sensitive to unknown tokens. Consider the sequence Harry 348.2598781 London, where the second token represents a phone number. Although the correct label is unknown to the model shown in Fig. 16, the latter will still try to assign a known label to the token. As a consequence, the whole sequence will result in a low fitting. It is worth noticing, however, that some regularities in the structure (e.g., the high probability that a sequence starts with a name) would allow to correctly classify the "known part" of the sequence, by avoiding to classify the "unlikely" term.

Recently, emphasis has been paid to the analysis of token context (i.e., of the tokens following and preceding the one at hand) for more accurate reconciliation. In particular, Maximum Entropy Markov Models (MEMMs) [19], i.e., conditional models that represent the probability of reaching a state given an observation and the previous state, can be seen as an attempt at contextualizing token reconciliation. However, MEMMs suffer from the well known label-bias problem [14].

Conditional random fields (CRFs) [14,18] are a probabilistic framework, that can be employed for text labeling and segmentation. The underlying idea is to define a conditional probability distribution over label sequences, given a particular observation sequence, rather than a joint distribution over both label and observation sequences. CRFs provide two major advantages. First, their conditional nature relaxes the strict independence assumptions required by HMMs to guarantee tractable inference. Second, CRFs avoid the label bias problem. Still, such improvements in the HMM technology represent global classification models, since they tend to classify each term into the sequence under consideration, and hence do not prevent the problem of misclassifying unknown tokens.

An unsupervised approach to text reconciliation is introduced in [2]. The basic idea here is to exploit *reference relations* for building segmentation models. The notion of reference relation denotes a collection of structured tuples, that are specific to a domain of interest and exemplify clean records for that domain. The approach consists of a two-step process. Assume that R is a reference relation with an attribute schema $A_1, \ldots, A_n$. Each column of R is considered as a dictionary of basic values for the corresponding attribute. Initially, a preprocessing step is performed for building an *attribute recognition model* (ARM) for each attribute of the reference relation schema. The generic ARM_i is a HMM that allows the evaluation of the probability with which a subsequence of tokens in an input string belongs to the domain of the corresponding schema attribute A_i. The ARMs of all attributes can be exploited to determine the best segmentation of an input string at the second (run-time) step. This involves to first learn the total order of attributes from a batch of input strings and to subsequently segment the individual input strings with respect to the detected attribute order. More specifically, the identification of a total attribute order requires the previous computation of pairwise precedence probabilities. These are probabilistic estimates of precedences between all pairs of attributes, that are provided by their corresponding *ARMs*. A total ordering among all of the attributes is hence discovered by choosing the best sequence of attributes, i.e., the sequence that maximizes the product of precedence probabilities of consecutive attributes with respect to the given order. Finally, an exhaustive search is employed to determine the best segmentation of an input string s into n token subsequences $s_1, \ldots, s_n$, such that the reconciliation of each s_i with the corresponding schema attribute A_{s_i} maximizes the overall reconciliation quality $\prod_{i=1}^{n} ARM_{s_i}(s_i)$ among all possible segmentations.

Notice that the exploitation of reference tables is a natural way of automatically building training sets for the text reconciliation problem described beforehand. And indeed, although declared as an unsupervised approach, this technique suffers from two general weaknesses, that are inherent of supervised methods. Foremost, a reference relation may not exist for a particular applicative scenario. Also, whenever the overall number of tuples involved is not sufficiently large, the columns of the employed relations may not be adequately rich dictionaries of basic domain tokens. This would affect the construction of *ARMs* and, hence, the overall segmentation effectiveness.

As to a more specific comparison with our contribution, the reference table approach [2] requires to initially learn the order with which attributes appear within the input data. By contrast, though being a supervised approach, RecBoost does not rely on learning attribute order from training data. This is due to the adoption of classification rules, that allow the

reconciliation of a given token on the sole basis of the relationships among the entities (i.e., further textual tokens, ontological categories and attributes) in the context surrounding the token at hand. Moreover, segmentation with respect to a given attribute order relies on the underlying assumption that such an ordering is fixed across input sequences. This may make reconciliation problematic when, instead, the tokens of two or more attribute values are interleaved (rather than being concatenated) in the data to segment.

Furthermore, the reference table approach adopts ARMs for reconciliating individual attribute values. However, ARMs are basically HMMs and, hence, suffer from the aforementioned limitations. Roughly speaking, ARMs are global classification models and, hence, overly specific in attribute recognition as far as three aspects are concerned, namely positional, sequential and token specificity. These aspects impose suitable generalizations for the ARMs: the adoption of a fixed three-layered topology capable of dealing with positional and sequential specificities and the exploitation of token hierarchies for mitigating token specificity. On the contrary, RecBoost relies on association rules for attribute value reconciliations. Association rules are better suited at detecting local patterns, especially when the underlying data to segment contain many contrasting specificities. Moreover, a natural generalization of classifiers, i.e., the improvement of their classification accuracy, is trivially obtained by attempting to reduce classifier complexity, via attribute and rule pruning.

7 Discussion and future works

The contribution of this paper was 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. Also, from a comparative analysis with state-of-the-art alternative approaches reveals the following two main arguments in favor of Recboost.

– Due to the variable number of classification stages, RecBoost gives the user better control over the trade-off between accuracy (i.e., the proportion of correctly classified tokens w.r.t. the classification behaviour of the overall RecBoost system) and recall (i.e., the proportion of correctly classified tokens w.r.t. the actual tokens to reconcile). In practice, the user can choose a classifier with a trade-off satisfying the requirements of the specific application.
– The generic RecBoost classifier is easier to interpret than existing methods such as DATAMOLD [3] and Mallet [18], since it produces symbolic rules using vocabulary from a domain-specific ontology.

There are some directions that are worth further research. First, notice that the proposed methodology is, in some sense, independent from the underlying rule-generation strategy. In this respect, it is interesting to investigate the adoption of alternative strategies for learning local classification models. This line is also , is a correlated with the effort for identifying a fully-automated technique for setting the parameters of *progressive classification*, in terms of required classification stages. Since parameters are model-dependent, two alternate strategies can be either to investigate different, parameter-free models, or to detect ways to enable a natural way of fixing the parameters of the system, on the basis of the inherent features of the text at hand, rather than relying on pre-specified estimates. The experimental section already

contains some pointers in the latter direction: however, more robust methods need in-depth investigation.

In addition, 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. 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. Adelberg B (1998) NoDoSE: A tool for semi-automatically extracting semistructured data from text documents. In: Haas LM, Tiwary A (eds) Proceedings of 1998 ACM SIGMOD conference on management of data. ACM Press, Seattle, WA, USA, June 1998, pp 283–294
2. Agichtein E, Ganti V (2004) Mining reference tables for automatic text segmentation. In: Kim W, Kohavi R, Gehrke J, DuMouchel W (eds) Proceedings of 2004 ACM SIGKDD conference on knowledge discovery and data mining. ACM Press, Seattle, WA, USA, August 2004, pp 20–29
3. Borkar VR, Deshmukh K, Sarawagi S (2001) Automatic segmentation of text into structured records. In: Aref WG (ed) Proceedings of 2001 ACM SIGMOD conference on management of Data. ACM Press, Santa Barbara, CA, USA, May 2001, pp 175–186
4. Brill E (1995) Transformation-based error-driven learning and natural language processing: a cased study in POS tagging. Comput Linguist 21(4):543–565
5. Califf ME, Mooney RJ (1999) Relational learning of pattern-match rules for information extraction. In: Proceedings of 16th national conference on artificial intelligence. AAAI/MIT Press, Madison, WI, USA, July 1999, pp 328–334
6. Cohen WW (1995) Learning to classify english text with ILP methods. In: De Raedt L (ed). Proceedings of the 5th international Workshop on inductive logic programming. Katholieke Universiteit Leuven, Haverlee, Belgium, pp 3–24
7. Elmagarmid AK, Panagiotis GI, Verykios VS (2007) Duplicate Record Dectection: A Survey. IEEE Trans Knowl Data Eng 19(1):1–16
8. Flesca F, Manco G, Masciari E (2004) Web wrapper induction: a brief survey.. AI Commun 17(2):57–61
9. Freitag D (1998) Toward general-purpose learning for information extraction. In: Proceedings of 17th national conference on computational linguistics. ACL/Morgan Kaufmann Publishers, Universit de Montral, Montreal, Quebec, Canada, August 1998, pp 404–408
10. Gu L, Baxter RA, Vickers D et al (2003) Record linkage: current practice and future directions. Technical report. CSIRO Mathematical and Information Sciences, Australia
11. Hernández MA, Stolfo J (1998) Real-world data is dirty: data cleansing and the merge/purge problem. Data Mining Knowl Discov 2(1):9–37
12. Junker M, Sintek M, Rinck M (1999) Learning for text categorization and information Extraction with ILP. In: Cussens J, Dzeroski S (eds) Learning language in logic. Springer Heidelberg, pp 247–258
13. Kupiec J (1992) Robust part-of-speech tagging using a hidden Markov model. Comput Speech Lang 6(3):225–242
14. Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Brodley CE, Pohoreckyj Danyluk A (eds). Proceedings of 18th international conference on machine learning. Morgan Kaufmann, Williamstown, MA, USA, June 2001, pp 282–289
15. Liu B, Hsu W, Ma Y (1998) Integrating classification and association rule mining. In: Agrawal R, Stolorz PE, Piatetsky-Shapiro G (eds) Proceedings of 4th ACM SIGKDD international conference on knowledge discovery and data mining. AAAI Press, New York City, NY, USA, August 1998, pp 80–86
16. Manning CD, Schultze C (1999) Foundations of statistical natural language processing. MIT Press, Cambridge
17. Marquez L, Padro L, Rodriguez H (2000) A machine learning approach to POS tagging. Mach Learn 39(1):59–91
18. McCallum A (2002) MALLET: a machine learning for language toolkit. http://mallet.cs.umass.edu
19. McCallum A, Freitag D, Pereira F (2000) Maximum entropy Markov models for information extraction and segmentation. In: Langley P (ed) Proceedings of 17th international conference on machine learning. Morgan Kaufmann, Standford University, Standord, CA, USA, June 2000, pp 591–598

20. Mukherjee S, Ramakrishnan IV (2004) Taming the unstructured: creating structured content from partially labeled schematic text sequences. In: Meersman R, Tari Z (eds) Proceedings of 12th CoopIS/DOA/ODBASE international conference. Springer, Agia Napa, Cyprus, October 2004, pp 909–926
21. Soderland S (1999) Learning information extraction rules for semi/structured and free text. Mach Learn 34:233–272
22. Srikant R, Agrawal R (1995) Mining generalized association rules. In: Dayal U, Gray PMD, Nishio S (eds) Proceedings of 21th international conference on Very large databases. Morgan Kaufmann, Zurich, Switzerland, September 1995, pp 407–419
23. Winkler WE (1999) The state of record linkage and current research problems. Technical report. Statistical Research Division, U.S. Census Bureau, Wachington, DC

Authors Biography

Eugenio Cesario is a temporary researcher at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy. He received his Ph.D. in Systems and Computer Engineering from the University of Calabria in 2006. From 2003 to 2006 he was a research fellow, at ICAR-CNR, working on data mining systems and applications. His research interests are currently focused on distributed data mining, Grid programming environments and Grid services architectures.

Francesco Folino is currently research fellow at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy. He graduated in Computer Science Engineering in 2003, and holds a Ph.D. in Computer Science Engineering from the University of Calabria (Italy) in 2006. From 2003 to 2006 he was a Ph.D. student at University of Calabria. His research interests are focused on knowledge discovery and data mining; Data Warehousing and OLAP. He is involved in several projects at ICAR-CNR, concerning applications of data mining and knowledge discovery.

Antonio Locane currently works at Exeura, a spin-off company of the University of Calabria (Italy) operating in the Knowledge Management area. He graduated in Computer Science Engineering in 2004 and, since October 2005, he is a Ph.D. Student at University of Calabria. He was research fellow at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy from 2004 to 2006. His research activity mainly concerns with data mining query languages and schema reconciliation for warehousing and mining.

Giuseppe Manco is currently senior researcher at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy, and contract professor at University of Calabria, Italy. He graduated in Computer Science summa cum laude, in 1994, and holds a Ph.D. in Computer Science from the University of Pisa. He has been contract researcher at the CNUCE Institute in Pisa, Italy, and visiting fellow at the CWI Institute in Amsterdam, Nederlands. His current research interests include deductive databases; knowledge discovery and data mining; Web databases and semistructured data.

Riccardo Ortale is researcher at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy, and contract professor at University of Calabria. He graduated in Computer Science Engineering summa cum laude from the University of Calabria. He has a master's degree in Internet Software Design from Cefriel-Politecnico di Milano in collaboration with Siemens SBS and holds a Ph.D. in Computer Science and System Engineering from the University of Calabria. His current research interests include knowledge discovery and data mining; Web databases and semistructured data; Web personalization.