Learning to Order: A Relational Approach

Donato Malerba and Michelangelo Ceci

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona, 4 - 70126 Bari - Italy
{malerba,ceci}@di.uniba.it

Abstract. In some applications it is necessary to sort a set of elements according to an order relationship which is not known *a priori*. In these cases, a training set of ordered elements is often available, from which the order relationship can be automatically learned. In this work, it is assumed that the correct succession of elements in a training sequence (or chain) is given, so that it is possible to induce the definition of two predicates, $first/1$ and $succ/2$, which are then used to establish an ordering relationship. A peculiarity of this work is the relational representation of training data which allows various relationships between ordered elements to be expressed in addition to the ordering relationship. Therefore, an ILP learning algorithm is applied to induce the definitions of the two predicates. Two methods are reported for the identification of either single chains or multiple chains on new objects. They have been applied to the problem of learning the reading order of layout components extracted from document images. Experimental results show the effectiveness of the proposed solution.

1 Introduction

Many applications require sorting a set of elements according to either a partial or a total ordering relationship. The problem can be efficiently solved by applying sorting algorithms when the ordering relationship is known *a priori*, but there are cases in which no definition of ordering relationship is available due to several difficulties in formalizing one. A prominent example is represented by preference functions, which indicate whether one element should be ranked before another. A preference function is typically subjective and difficult to elicit, although it can be relatively easy to collect instances of ordered elements from which the preference function can be automatically induced. Another example is the reading order of layout components in a page. The rule of thumb "Western-style documents are usually read top-bottom and left-right" is an ambiguous statement, which is not appropriate for many newspapers and magazines. Also in this case, it is easy to collect examples of correct reading sequences from which the reading order rules specific to a class of documents can be learned.

The problem of learning how to construct an ordering, given a collection of instances of ordered elements, has been faced by Cohen *et al* [6] who propose a two-stage approach. In the first stage (*learning*), a binary preference function $PREF(u, v)$ is learned, which indicates how certain it is that u should be

Z.W. Raś, S. Tsumoto, and D. Zighed (Eds.): MCD 2007, LNAI 4944, pp. 209–223, 2008.

ranked before v. In the second stage (*sorting*), new instances are ordered so as to maximize the agreement with the learned preference function. The function $PREF(u, v)$ is a linear combination of primitive preference functions, each of which is associated with a ranking expert, and the learning process consists in defining a weight for each primitive preference function on the basis of training sets of ordered pairs (u, v). Kamishima and Akaho [13] propose a naive Bayes approach to estimate $PREF(u, v)$ and compare two alternative optimality criteria (sum and product of values taken by PREF) to be maximized in the sorting stage.

In both works, an ordered pair (u, v) in the training set is interpreted as v is ranked above u, and the learning task aims to induce a definition of an ordering which is consistent with input ordered pairs. However, in many applications the ordered pair (u, v) in the training set can be interpreted as "v is the successor of u", in which case the learning task can be slightly different, namely learning the definition of the *successor* relationship between elements. Once the successor relationship is learned, an ordering relationship can be established.

In this paper, we follow this approach to learning how to order elements. Given both positive and negative instances of two predicates, namely $first/1$ and $succ/2$, we first induce the definitions of "first element" and "successor element" in a sequence (or *chain*) of elements, and then we apply these definitions to a new set of elements, in order to reconstruct a possible partial or total ordering of these elements. The main advantage of this approach is that it can also be applied to those tasks characterized by the following properties: a) not all elements have to be ordered - only those involved in a direct succession relationship; b) different sequences can be defined on subsets of elements. Two examples of these tasks are the detection of the reading order between document layout components [2] and the design of workflows from process logs [20].

The task considered in this paper is *predictive* and differs from another *descriptive* task reported in the works [19], where the problem is discovering fragments of order, and [10], where the problem is that of describing a set of sequences by a single (or a set of) partial orders occurring in the sequences.

Another important difference with respect to related works is the representation of training data. In all previous works, the ordering relationship is the only one considered between elements, which are represented as rows of a single table, whose columns correspond to attributes of the elements. However, in several applications this representation is quite restrictive. For instance, in reading order detection some spatial relationships can be defined between layout components and the reading order can actually depend on these spatial relationships (e.g., the next layout component is 'below' the one currently read). To consider these additional relationships we resort to a *relational* representation with several tables which describe possibly different types of elements and the various relationships between them. Therefore, a peculiarity of this work is that training data are *complex objects* and that ordered elements are *basic components* of these complex objects.

The relational representation of complex objects requires the application of inductive logic programming (ILP) [23,14,24] methods in order to induce a

definition of the predicates $first/1$ and $succ/2$. In this work, we resort to the application of the ILP system ATRE [17] to learn a logical theory which defines the two predicates and is then used to reconstruct a partial or total ordering in new structured complex objects.

The paper is organized as follows. The problem of learning to order objects is formally defined in Section 2. The machine learning system ATRE, applied to the problem of learning the logical theory, is introduced in Section 3, while the application of the learned theory in order to reconstruct a partial order relationship, is reported in Section 4. Finally, the application to the document image processing domain is illustrated in Section 5, where experimental results are also reported and commented.

2 Problem Definition

In order to formalize the learning problem, some useful definition are necessary.

Definition 1 (*Partial Order [11]***).** *Let A be a set of basic components of a complex object, a partial order P over A is a relation $P \in A \times A$ such that P is*

1. *reflexive* $\forall s \in A \Rightarrow (s, s) \in P$
2. *antisymmetric* $\forall s_1, s_2 \in A$*:* $(s_1, s_2) \in P \wedge (s_2, s_1) \in P \Leftrightarrow s_1 = s_2$
3. *transitive* $\forall s_1, s_2, s_3 \in A$*:* $(s_1, s_2) \in P \wedge (s_2, s_3) \in P \Rightarrow (s_1, s_3) \in P$

When P satisfies the antisymmetric, the transitive and the irreflexive ($\forall s \in A \Rightarrow (s, s) \notin P$) properties, it is called a weak partial order over A.

Definition 2 (*Total Order***).** *Let A be a set of basic components of a complex object, a partial order T over the set A is a total order iff* $\forall s_1, s_2 \in A$*:* $(s_1, s_2) \in T \vee (s_2, s_1) \in T$

Definition 3 (*Complete chain, Chain reduction***).** *Let A be a set of basic components of a complex object, let D be a weak partial order over A, let $B = \{a \in A | (\exists b \in A \text{ s.t. } (a, b) \in D \vee (b, a) \in D)\}$ be the subset of elements in A related to any element in A itself. If $D \cup \{(a, a) | a \in B\}$ is a total order over B then D is a complete chain over A.*

Furthermore, $C = \{(a, b) \in D | \neg \exists c \in A \text{ s.t. } (a, c) \in D \wedge (c, b) \in D\}$ is the reduction of the chain D over A.

Example 1. Let $A = \{a, b, c, d, e\}$. $D = \{(a, b), (a, c), (a, d), (b, c), (b, d), (c, d)\}$ is a complete chain over A, then $C = \{(a, b), (b, c), (c, d)\}$ is its reduction.

Indeed, for our purposes it is equivalent to deal with complete chains or their reduction. Henceforth, for the sake of simplicity, the term *chain* will denote the reduction of a complete chain. By resorting to definitions above, it is possible to formalize the ordering induction problem as follows:

Given

- A description $DesTO_i$ in the language L of the set of n training complex objects $TrainingObjs = \{TP_i \in \Pi | i = 1..n\}$ (where Π is the set of complex objects).
- A description $DesTC_i$ in the language L of the set TC_i of chains (over $TP_i \in TrainingObjs$) for each $TP_i \in TrainingObjs$.

***Find*:** An intensional definition T in the language L of a chain over a generic compex object $O \in \Pi$ such that T is complete and consistent with respect to all training chains descriptions $DesTC_i$, $i = 1..n$.

In this problem definition, we refer to the intensional definition T as a first order logic theory, that is, a set of first order definite clauses [16]. The fact that T is complete and consistent with respect to all training chains descriptions can be formally described as follows:

Definition 4 (Completeness and Consistency). *Let*

- *T be a logic theory describing chains instances expressed in the language L.*
- *E^+ be the set of positive examples for the chains instances. ($E^+ = \bigcup_{i=1..n}(\bigcup_{TC \in TC_i} TC)$).*
- *E^- be the set of negative examples for the chains instances. ($E^- = \bigcup_{i=1..n}(TP_i \times TP_i)/E^+$).*
- *$DesE^+$ be the description of E^+ in L.*
- *$DesE^-$ be the description of E^- in L.*

then T is complete and consistent with respect to all training chains descriptions iff $T \models DesE^+ \ \wedge \ T \not\models DesE^-$

This formalization of the problem permits to represent and identify distinct orderings on the same complex object and allows to avoid to include in the ordering basic components that should not be included.

3 Learning the Logical Theory with ATRE

ATRE is an ILP system that can learn recursive theories from examples. The learning problem solved by ATRE can be formulated as follows:

Given

- a set of *concepts* $K_1, K_2, \ldots, K_r$ to be learned
- a set of *observations* O described in a language $\mathcal{L}_O$
- a *background theory* BK described in a language $\mathcal{L}_{BK}$,
- a *language* of hypotheses $\mathcal{L}_H$ which defines the space of hypotheses S_H
- a user's *preference criterion* PC

Find

A logical theory $T \in S_H$, which defines the concepts K_1, K_2, ..., K_r, such that T is complete and consistent with respect to O and satisfies the preference criterion PC.

The *completeness* property holds when the theory T explains all observations in O of the r concepts K_1, K_2, ..., K_r, while the *consistency* property holds when the theory T explains no counter-example in O of any concept C_i. The satisfaction of these properties guarantees the correctness of the induced theory, with respect to the given observations O. The selection of the "best" theory is made on the basis of an inductive bias embedded in some heuristic function or expressed by the user of the learning system (preference criterion).

In the context of the ordering induction problem, we identified two concepts to be learned, namely $first/1$ and $succ/2$. The former refers to the the first basic component of a chain, while the latter refers to the relation *successor* between two basic components in a chain. By combining the two concepts it is possible to identify a partial ordering of basic components of a complex object.

Both the language of hypotheses $\mathcal{L}_H$ and the language of background knowledge $\mathcal{L}_{BK}$ are limited to linked, range-restricted definite clauses [7], whose literals can be of the two distinct forms:

$f(t_1, \ldots, t_n) = \text{Value}$ (simple literal)
$f(t_1, \ldots, t_n) \in \text{Range}$ (set literal),

where f and g are function symbols called *descriptors*, t_i's are *terms* (constants or variables) and *Range* is a closed interval of possible values taken by f.

Observations are represented as ground multiple-head clauses, called *objects*, which have a conjunction of simple literals in the head. Each object is associated with a unique object identifier (OID). The notion of multiple-head clauses in ATRE adapts the notion of *interpretation*, which is common to many relational data mining systems for efficiency reasons [8]. ATRE distinguishes objects from *examples*, which are described as pairs $\langle L, OID \rangle$, where L is a literal in the head of the object identified by *OID*. Examples can be either *positive* or *negative*.

At the high-level ATRE implements a *sequential covering* algorithm [22]. A recursive theory T is built iteratively, starting from an empty theory T_0, and then adding a new clause at each iteration. In this way we obtain a sequence of theories:

$$T_0 = \emptyset, T_1, \ldots, T_i, T_{i+1}, \ldots, T_n = T$$

such that $T_{i+1} = T_i \cup \{C\}$ for some clause C and $LHM(T_i) \subseteq LHM(T_{i+1})$, where $LHM(T)$ denotes the least Herbrand model of a theory T [16]. Let $pos(LHM(T))$ and $neg(LHM(T))$ be the number of positive and negative examples in $LHM(T)$, respectively. If we guarantee that:

1. $pos(LHM(T_i)) < pos(LHM(T_{i+1}))$, for each $i \in \{0, 1, \ldots, n-1\}$ and
2. $neg(LHM(T_i)) = 0$, for each $i \in \{0, 1, \ldots, n\}$,

then, after a finite number of iterations, a theory T, which is complete and consistent, is built. The first condition is guaranteed by selecting a positive example (or *seed*) $e^+ \notin LHM(T_i)$ of a concept K_j to be learned, and then by looking for a clause C, if any, such that $e^+ \in LHM(T_i \cup \{C\})$ (i.e., $pos(LHM(T_i \cup \{C\})) > pos(LHM(T_i))$). The second condition is more difficult to guarantee since the addition of a locally consistent clause C to a theory T_i does not preserve consistency of $T_i \cup \{C\}$ (non-monotonicity of the normal ILP setting). The approach followed in ATRE consists of simple syntactic changes in T_i, which eventually creates new *layers* [17].

The automated discovery of dependencies between concepts K_1, K_2, ..., K_r is based on a variant of the sequential covering learning strategy, which is traditionally adopted by single concept learning systems that generate clauses with the same literal in the head at each iteration. In multiple concept learning, clauses generated at each iteration may refer to different concepts. In addition, the body of the clause generated at the i-th iteration may involve any concept K_1, K_2, ..., K_r for which at least a clause has been added to the theory partially learned in previous iterations. In this way, dependencies between concepts can be generated.

At each iteration of the main loop of the sequential covering algorithm, clauses for distinct concepts are generated, and then one of them is picked. Since the generation of a clause depends on a seed, several seeds have to be chosen (if any, at least one seed per concept to be learned). Therefore, the search space is a forest of as many search-trees as the number of chosen seeds. Each search-tree is rooted with a unit clause and ordered by the generalization model adopted in ATRE (*generalized implication* [17]). The forest can be processed in parallel by as many concurrent tasks as the number of search-trees. Each task traverses the specialization hierarchy top-down through a sequential covering strategy, but synchronizes traversal with the other tasks at each level. Search proceeds toward deeper and deeper levels of the specialization hierarchies until at least a user-defined number of consistent clauses is found. Task synchronization is performed after that all "relevant" clauses at the same depth have been examined. A supervisor task decides whether the search should carry on or not on the basis of the results returned by the concurrent tasks. When the search is stopped, the supervisor selects the "best" consistent clause according to the user's preference criterion. This *separate-and-parallel-conquer* search strategy provides us with a solution to the problem of interleaving the induction of distinct concept definitions.

4 Application of the Learned Logical Theory

Once the logical theory has been learned, they can be applied to new complex objects in order to generate a set of ground atoms such as: $\{first(0) = true, succ(0,1) = true, \ldots, succ(4,3) = true, \ldots\}$ which can be used to reconstruct chains of (possibly logically labelled) basic components. In our approach, we propose two different solutions: 1) Identification of multiple chains of basic components. 2) Identification of a single chain of basic components.

By applying the logical theory learned by ATRE, it is possible to identify:

- A *directed* graph $G = \langle V, E \rangle$[1] where V is the set of nodes representing all the basic components of a complex object and $E = \{(b_1, b_2) \in V^2 | succ(b_1, b_2) = true\}$
- A list of initial nodes $I = \{b \in V | first(b) = true\}$
- A list of final nodes $F = \{b_2 \in V - I | \forall b_1 \in V \ (b_1, b_2) \notin E\}$.

Both approaches make use of G, I and T in order to identify chains.

Multiple Chains Identification. This approach aims at identifying a (possibly empty) set of chains over the set of basic components of a complex object. It is based on two steps, the first of which aims at identifying the heads (first elements) of the possible chains, that is the set

$$Heads = I \cup \{b_1 \in V | \ \exists b_2 \in V \ (b_1, b_2) \in E \ \wedge \forall b_0 \in V \ (b_0, b_1) \notin E\}$$

This set contains both nodes for which $first$ is true and nodes which occur as a first argument in a true *succ* atom and do not occur as a second argument in any true *succ* atom.

Once the set $Heads$ has been identified, it is necessary to reconstruct the distinct chains. Intuitively, each chain is the list of nodes forming a path in G which begins with a node in $Heads$ and ends with a node without outgoing edges. Formally, an extracted chain $C \subseteq E$ is defined as follows:

$$C = \{(b_1, b_2), (b_2, b_3), \ldots, (b_k, b_{k+1})\}$$

such that

1. $b_1 \in Heads$,
2. $b_{k+1} \in F$,
3. $\forall i = 1..k : (b_i, b_{i+1}) \in E$
4. $\forall i = 1..k \forall j = i + 1..k + 1 b_i \neq b_j$.

The last condition imposes that the same node cannot appear more than once in the same chain In order to avoid cyclic paths.

Single Chain Identification. The result of the second approach is a single chain. Following the proposal reported in [6], we aim at iteratively evaluating the most promising node to be appended to the resulting chain. More formally, let $PREF_G : V \times V \rightarrow \{0, 1\}$ be a preference function defined as follows:

$$PREF_G(b_1, b_2) = \begin{cases} 1 \text{ if } b_1 = b_2 \text{ or a path connecting } b_1 \text{ and } b_2 \text{ exists in } G \\ 0 \text{ otherwise} \end{cases}$$

Let $\mu : V \rightarrow \mathbb{N}$ be the function defined as follows:

$$\mu(L, G, I, b) = countConnections(L, G, I, b) + outGoing(V/L, b) \\ -inComing(V/L, b)$$

[1] G is not a direct acyclic graph (dag) since it could also contain cycles.

where

- $G = \langle V, E \rangle$ is the ordered graph
- L is a list of *distinct* nodes in G
- $b \in V/L$ is a candidate node
- $countConnections(L, G, I, b) = |\{d \in L \cup I | PREF_G(d, b) = 1\}|$ counts the number of nodes in $L \cup I$ from which b is reachable.
- $outGoing(V/L, b) = |\{d \in V/L | PREF_G(b, d) = 1\}|$ counts the number of nodes in V/L reachable from b.
- $inComing(V/L, b) = |\{d \in V/L | PREF_G(d, b) = 1\}|$ counts the number of nodes in V/L from which b is reachable.

Algorithm 1 fully specifies the method for the single chain identification. The rationale is that at each step a node is added to the final chain. Such a node is that for which μ is the highest. Higher values of μ are given to nodes which can be reached from I, as well as from other nodes already added to the chain, and have a high (low) number of outgoing (incoming) paths to (from) nodes in V/L. Indeed, the algorithm returns an ordered list of nodes which could be straightforwardly transformed into a chain.

Algorithm 1 Single chain identification algorithm

```
1: findChain (G =< V, E >, I) Output: L: chain of nodes
2: L← ∅;
3: repeat
4:    best_mu ← −∞;
5:    for all  b ∈ V/L do
6:       cc ← countConnections(L, G, I, b);
7:       inC ← incoming(V/L, b); outG ← outGoing(V/L, b);
8:       if ((cc ≠ 0) OR (inC ≠ 0) OR (outG ≠ 0)) then
9:          μ ← cc + outG − inC;
10:         if best_mu < μ then
11:            best_b ← b; best_mu ← μ;
12:         end if
13:      end if
14:   end for
15:   if (best_mu <> −∞) then
16:      L.add(best_b);
17:   end if
18: until best_mu = −∞
19: return L
```

5 The Application: Learning Reading Order of Layout Components

In this paper, we investigate an application to the document image understanding problem. More specifically, we are interested in determining the reading order of layout components in each page of a multi-page document. Indeed, the spatial

order in which the information appears in a paper document may have more to do with optimizing the print process than with reflecting the logical order of the information contained. Determining the correct reading order can be a crucial problem for several applications. By following the reading order recognized in a document image, it is possible to cluster together text regions labeled with the same logical label into the same textual component (e.g., "introduction", "results", "method" of a scientific paper). In this way, the reconstruction of a single textual component is supported and advanced techniques for text processing can be subsequently applied. For instance, information extraction methods may be applied locally to reconstructed textual components. Moreover, retrieval of document images on the basis of their textual contents is more effectively supported.

Several works on reading order detection have already been reported in the literature [26][12][21][25][1] [4]. A common aspect of all methods is that they strongly depend on the specific domain and are not "reusable" when the classes of documents or the task at hand change. There is no work, to the best of our knowledge, that handles the reading order problem by resorting to machine learning techniques, which can generate the required knowledge from a set of training layout structures whose correct reading order has been provided by the user. In previous works on document image understanding, we investigated the application of machine learning techniques to several knowledge-based document image processing tasks, such as classification of blocks [3], automatic global layout analysis correction [18], classification of documents into a set of pre-defined classes and logical labelling. Following this mainstream of research, herein we consider the problem of learning the reading order.

In this context the limitations posed by the single table assumption are quite restrictive for at least two reasons. First, layout components cannot be realistically considered independent observations, because their spatial arrangement is mutually constrained by formatting rules typically used in document editing. Second, spatial relationships between a layout component and a variable number of other components in its neighborhood cannot be properly represented by a fixed number of attributes in a table. Even more so, the representation of properties of the other components in the neighborhood, because different layout components may have different properties (e.g., the property "brightness" is appropriate for half-tone images, but not for textual components). Since the single-table assumption limits the representation of relationships (spatial or non) between examples, it also prevents the discovery of this kind of pattern, which can be very useful in document image mining.

For these reasons, the ILP approach proposed in this paper seems to be appropriate for the task at hand. In ATRE, training observations are represented by ground multiple-head clauses [15], called *objects*, which have a conjunction of simple literals in the head. The head of an object contains positive and negative examples for the concepts to be learned, while the body contains the description of layout components on the basis of geometrical features (e.g. width, height) and topological relations (e.g. vertical and horizontal alignments) existing among

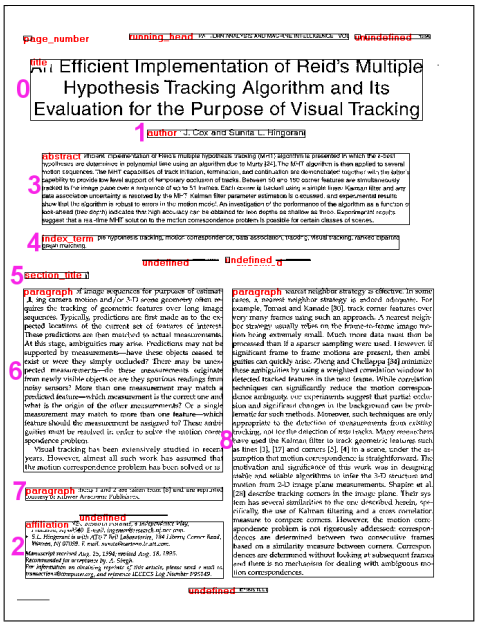

Fig. 1. A document page: the input reading order chain. Sequential numbers indicate the reading order.

blocks, the type of the content (e.g. text, horizontal line, image) and the logic type of a block (e.g. title or authors of a scientific paper). Terms of literals in objects can only be constants, where different constants represent distinct layout components within a page. An example of object description generated for the document page in Figure 1 is the following:

$$
\begin{aligned}
&object('tpami17\_1\text{-}13', [class(p) = tpami,\\
&first(0) = true, first(1) = false, ...\\
&succ(0,1) = true, succ(1,2) = true, ..., succ(7,8) = true, succ(2,10) = false, ...],\\
&[part\_of(p,0) = true, ...,\\
&height(0) = 83, height(1) = 11, ..., width(0) = 514, width(1) = 207, ...,\\
&type\_of(0) = text, ..., type\_of(11) = hor\_line,\\
&title(0) = true, author(1) = true, affiliation(2) = true, .., undefined(16) = true,\\
&x\_pos\_centre(0) = 300, x\_pos\_centre(1) = 299, ...,\\
&y\_pos\_centre(0) = 132, y\_pos\_centre(1) = 192, ...,\\
&on\_top(9,0) = true, on\_top(15,0) = true, ..., to\_right(6,8) = true, ...\\
&alignment(16,8) = only\_right\_col, alignment(17,5) = only\_left\_col, ...\\
&class(p) = tpami, page(p) = first]).
\end{aligned}
$$

The constant p denotes the whole page while the remaining integer constants (0, 1, ..., 17) identify distinct layout components. In this example, the block number 0 corresponds to the first block to read ($first(0) = true$), it is a

textual component ($type\_of(0) = text$) and it is logically labelled as 'title' ($title(0) = true$). Block number 1 (immediately) follows block 0 in the reading order ($succ(0, 1) = true$), it is a textual component and it includes information on the authors of the paper ($author(1) = true$).

As explained in the previous sections, ATRE learns a logical theory T defining the concepts $first/1$ and $succ/2$ such that T is complete and consistent with respect to the examples. This means that it is necessary to represent both positive and negative examples and the representation of negative examples for the concept $succ/2$ poses some feasibility problems due to their quadratic growth. In order to reduce the number of negative examples, we resort to sampling techniques. In our case, we sampled negative examples by limiting their number to 1000% of the number of positive examples. This way, it is possible to simplify the learning stage and to have a logical theory which is less specialized and avoids overfitting.

In order to evaluate the applicability of the proposed approach to reading order identification, we considered a set of multi-page articles published in an international journal. In particular, we considered twenty-four papers, published as either regular or short articles, in the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), in the January and February issues of 1996. Each paper is a multi-page document; therefore, we processed 211 document images. Each document page corresponds to a 24bit TIFF color image.

Initially, document images are pre-processed in order to segment them, perform layout analysis, identify the membership class and map the layout structure of each page into the logical structure. Training examples are then generated by manually specifying the reading order. In all, 211 positive examples and 3,263 negative examples for the concept $first/1$ and 1,418 positive examples and 15,518 negative examples for the concept $succ/2$ are generated. The average number of layout components in training chains is about 8.0.

We evaluated the performance of the proposed approach by means of a six-fold cross-validation: the dataset is first divided into six folds of equal size and then, for every fold, the learner is trained on the remaining folds and tested on it.

For each learning problem, statistics on precision and recall of the learned logical theory are recorded. In order to evaluate the ordering returned by the proposed approach, we resort to metrics used in information retrieval to evaluate the returned ranking of results [9]. Herein we consider the metrics valid for partial orders evaluation.

In particular, we consider the *normalized Spearman footrule distance* which, given two complete lists L and L_1 on a set S (that is, L and L_1 are two different permutations without repetition of all the elements in S), is defined as follows:

$$F(L, L_1) = \frac{\sum_{b \in S} abs(pos(L, b) - pos(L_1, b))}{|S|^2/2} \quad (1)$$

where the function $pos(L, b)$ returns the position of the element b in the ordered list L. $F(L, L_1)$ is always between 0 and 1, where 0 means that L and L_1 are exactly the same, while 1 means that the two lists are completely in the opposite order. In other terms, the lower the $F(L, L_1)$, the better.

The *induced footrule distance* $F(L|_{L_1}, L_1)$ is the variant of $F(L, L_1)$ computed by projecting the list L on L_1: it is used when L_1 is a sublist of L.

The normalized Spearman footrule distance can be straightforwardly generalized to the case of several lists:

$$F(L, L_1, \ldots, L_k) = 1/k \sum_{i=1\ldots k} F(L, L_i).$$

In order to consider partial (and not total) orders, we resorted to a variant called (*induced normalized footrule distance*):

$$F(L, L_1, \ldots, L_k) = 1/k \sum_{i=1\ldots k} F(L|_{L_i}, L_i)$$

Since this measure does not take into account the length of single lists, we also adopted the *normalized scaled footrule distance*:

$$F'(L, L_1) = \frac{\sum_{b \in S} abs(pos(L, b)/|L| - pos(L_1, b)/|L_1|)}{|L_1|/2} \quad (2)$$

Also in this case it is possible to extend the measure to the case of multiple lists: $F'(L, L_1, \ldots, L_k) = 1/k \sum_{i=1\ldots k} F'(L|_{L_i}, L_i)$.

In this study, we apply such distance measures to chains. In particular, both FD= $F(L|_{L_1}, L_1)$ and SFD=$F'(L|_{L_1}, L_1)$ are used in the evaluation of single chain identification, while IFD=$F(L, L_1, \ldots, L_k)$ and ISFD=$F'(L, L_1, \ldots, L_k)$ are used in the evaluation of multiple chains identification.

Results reported in Table 1 show that the system has a precision of about 65% and a recall greater than 75%. Moreover, there is no significant difference in terms of recall between the two concepts, while precision is higher for the *succ* concept. This is mainly due to the specificity of the clauses learned for the concept $first$: clauses learned for the concept $first$ cover (on average) fewer positive examples than clauses learned for the concept *succ* (see Table 2). We can conclude that the concept $first$ appears to be more complex to learn than the concept *succ*, probably because of the lower number of training examples (one per page).

Experimental results concerning the reconstruction of single/multiple chains are reported in Table 3. We recall that the lower the distance value the better the reconstruction of the original chain(s). By comparing results in terms of

Table 1. Precision and Recall results shown per concept to be learned

Concept	$first/1$		$succ/2$		*Overall*	
	Precision %	Recall%	Precision%	Recall%	Precision%	Recall%
FOLD1	75.00	50.00	76.90	64.10	76.60	61.80
FOLD2	66.70	63.20	74.10	65.20	73.00	64.90
FOLD3	74.30	78.80	81.00	66.10	80.10	67.40
FOLD4	69.40	71.40	67.80	56.30	68.00	58.20
FOLD5	66.70	66.70	78.40	68.70	76.80	68.40
FOLD6	71.00	61.10	79.40	62.90	78.20	62.60
AVG	70.52	65.20	76.27	63.88	75.45	63.88

Table 2. Number of learned clauses per positive examples

Concept	$first\_to\_read/1$		$succ\_in\_reading/2$	
	No of clauses	Training POS exs	No of clauses	Training POS exs
FOLD1	42	175	162	1226
FOLD2	46	173	145	1194
FOLD3	42	178	149	1141
FOLD4	42	176	114	1171
FOLD5	40	178	166	1185
FOLD6	41	175	177	1173
AVG coverage	4.17		7.77	

Table 3. Reading order reconstruction results

Concept	Multiple chains		Single chain	
	AVG. IFD%	AVG. ISFD%	AVG. FD%	AVG. SFD%
FOLD1	13.18	21.12	47.33	10.17
FOLD2	10.98	18.51	46.32	8.13
FOLD3	1.31	26.91	47.32	17.63
FOLD4	1.32	24.00	49.96	14.51
FOLD5	0.90	22.50	49.31	10.60
FOLD6	0.90	27.65	54.38	12.97
AVG	4.76	23.45	49.10	12.33

the *footrule distance* measure (IFD vs FD), we note that the reconstruction of multiple chains shows better results than the reconstruction of single chains. Indeed, this result does not take into account the length of the lists. When considering the length of the lists (ISFD vs. SFD) the situation is completely different and the reconstruction of single chains outperforms the reconstruction of multiple chains.

Some examples of clauses learned by ATRE are reported below:

1. $first(X1) = true \leftarrow x\_pos\_centre(X1) \in [55..177]$, $y\_pos\_centre(X1) \in [60..121], height(X1) \in [98..138]$.
2. $first(X1) = true \leftarrow title(X1) = true,\ x\_pos\_centre(X1) \in [293..341]$, $succ(X1, X2) = true$.
3. $succ(X2, X1) = true \leftarrow affiliation(X1) = true,\ author(X2) = true$, $height(X1) \in [45..124]$.
4. $succ(X2, X1) = true \leftarrow alignment(X1, X3) = both\_columns$, $on\_top(X2, X3) = true, succ(X1, X3) = true,\ height(X1) \in [10..15]$

They show that ATRE is particularly indicated for the task at hand since it is able to identify dependencies among concepts to be learned or even recursion.

6 Conclusions

In this paper, we present an ILP approach to the problem of inducing a partial ordering between basic components of complex objects. The proposed solution is based on learning a logical theory which defines two predicates $first/1$ and $succ/2$. The learned theory should be able to express dependencies between

the two target predicates. For this reason we used the learning system ATRE which is able to learn mutually recursive predicate definitions. In the recognition phase, learned predicate definitions are used to reconstruct reading order chains according two different modalities: single vs. multiple chains identification.

The proposed approach can be applied to several application domains. In this paper, it has been applied to a real-world problem, namely detecting the reading order between layout components extracted from images of multi-page documents. Results prove that learned logical theories are quite accurate and that the reconstruction phase significantly depends on the application at hand. In particular, if the user is interested in reconstructing the actual chain (e.g. text reconstruction for rendering purposes), the best solution is in the identification of single chains. On the contrary, when the user is interested in recomposing text such that sequential components are correctly linked (e.g. in information extraction), the most promising solution is the identification of multiple chains.

For future work we intend to compare the logic-based approach proposed in this paper with a probabilistic-based approach reported in [5]. Moreover, we plan to extend our empirical investigation to other application domains as well as to synthetically generated datasets.

Acknowledgments

This work is in partial fulfillment of the research objectives of the project "D.A.M.A." (Document Acquisition, Management and Archiving). The authors also wish to thank Lynn Rudd for her help in reading the manuscript.

References

1. Aiello, M., Monz, C., Todoran, L., Worring, M.: Document understanding for a broad class of documents. International Journal on Document Analysis and Recognition-IJDAR 5(1), 1–16 (2002)
2. Aiello, M., Smeulders, A.: Bidimensional relations for reading order detection. In: Proceedings of Joint Conference on Information Science (2003)
3. Altamura, O., Esposito, F., Malerba, D.: Transforming paper documents into XML format with WISDOM++. International Journal on Document Analysis and Recognition-IJDAR 4(1), 2–17 (2001)
4. Breuel, T.M.: High performance document layout analysis. In: Proceedings of the 2003 Symposium on Document Image Understanding (SDIUT 2003) (2003)
5. Ceci, M., Berardi, M., Porcelli, G., Malerba, D.: A data mining approach to reading order detection. In: ICDAR 2007: 9th International Conference on Document Analysis and Recognition, pp. 924–928 (2007)
6. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research (JAIR) 10, 243–270 (1999)
7. De Raedt, L.: Interactive Theory Revision. Academic Press, London (1992)
8. Džeroski, S., Lavrač, N.: Relational Data Mining. Springer, Berlin (2001)
9. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: WWW 2001: Proceedings of the 10th international conference on World Wide Web, pp. 613–622. ACM Press, New York (2001)

10. Gionis, A., Kujala, T., Mannila, H.: Fragments of order. In: KDD 2003: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 129–136. ACM Press, New York (2003)
11. Grimaldi, R.P.: Discrete and Combinatorial Mathematics, an Applied Introduction, 3rd edn. Addison Wesley, Reading (1994)
12. Ishitani, Y.: Document transformation system from papers to XML data based on Pivot XML document method. In: ICDAR 2003: 7th International Conference on Document Analysis and Recognition, p. 250. IEEE Computer Society, Los Alamitos (2003)
13. Kamishima, T., Akaho, S.: Learning from order examples. In: Proceedings of the 2nd IEEE International Conference on Data Mining, pp. 645–648 (2002)
14. Lavrač, N., Džeroski, S.: Inductive Logic Programming: techniques and applications. Ellis Horwood, Chichester (1994)
15. Levi, G., Sirovich, F.: Generalized and/or graphs. Artificial Intelligenge 7(3), 243–259 (1976)
16. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer, Berlin (1987)
17. Malerba, D.: Learning recursive theories in the normal ILP setting. Fundamenta Informaticae 57(1), 39–77 (2003)
18. Malerba, D., Esposito, F., Altamura, O., Ceci, M., Berardi, M.: Correcting the document layout: A machine learning approach. In: ICDAR 2003: 7th International Conference on Document Analysis and Recognition, p. 97 (2003)
19. Mannila, H., Meek, C.: Global partial orders from sequential data. In: KDD 2000: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 161–168. ACM Press, New York (2000)
20. Maruster, L., Weijters, A., van der Aalst, W., van den Bosch, A.: Process mining: Discovering direct successors in process logs. In: Lange, S., Satoh, K., Smith, C.H. (eds.) DS 2002. LNCS, vol. 2534, pp. 364–373. Springer, Heidelberg (2002)
21. Meunier, J.-L.: Optimized xy-cut for determining a page reading order. In: ICDAR 2005: 8th International Conference on Document Analysis and Recognition, pp. 347–351. IEEE Computer Society, Washington (2005)
22. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
23. Muggleton, S.: Inductive Logic Programming. Academic Press, London (1992)
24. Nienhuys-Cheng, S.-W., de Wolf, R.: Foundations of inductive logic programming. Springer, Heidelberg (1997)
25. Taylor, S.L., Dahl, D.A., Lipshutz, M., Weir, C., Norton, L.M., Nilson, R., Linebarger, M.: Integrated text and image understanding for document understanding. In: HLT 1994: Proceedings of the workshop on Human Language Technology, pp. 421–426 (1994)
26. Tsujimoto, S., Asada, H.: Understanding multi-articled documents. In: Proceedings of the 10th International Conference on Pattern Recognition, pp. 551–556 (1990)