Incremental Learning of First Order Logic Theories for the Automatic Annotations of Web Documents

Floriana Esposito Stefano Ferilli Nicola Di Mauro
Teresa M.A. Basile
Università degli Studi di Bari-Dipartimento di Informatica
Via Orabona, 4 - 70126 Bari - Italy
{esposito, ferilli, ndm, basile}@di.uniba.it

Abstract

Organizing large repositories spread throughout the most diverse Web sites rises the problem of effective storage and efficient retrieval of documents. This can be obtained by selectively extracting from them the significant textual information, contained in peculiar layout components, that in turn depend on the identification of the correct document class. The continuous flow of new and different documents in a weakly structured environment like the Web calls for incrementality, as the ability to continuously update or revise a faulty knowledge previously acquired, while the need to express structural relations among layout components suggest the exploitation of a powerful and symbolic representation language. This paper proposes the application of incremental first-order logic learning techniques in the document layout preprocessing steps, supported by good results obtained in experiments on a real dataset.

1 Introduction & Motivations

Today, huge amounts of documents are spread throughout the most diverse Web sites. Up to now most research effort was spent on principles and techniques for setting up and managing Digital Libraries [1, 3, 7, 8, 9]. Thus, the next problem is developing techniques for semantics-related tasks, such as information retrieval and integration. Indeed, in order to capture the semantics underlying a document on the Web, one would need access to the conceptual model of the information source. Ontologies can hardly deal with heterogeneous representations in a poorly structured environment like the Web. A more effective indexing can be obtained by focusing on the most relevant text components in the document, that can be identified based on its logical layout structure. Performing such a task consists in recognizing the correct document class first, and then its significant components, from which extracting the relevant information. Since an important role to carry out this task is played by the relationships between components, a Document Image Understanding technique should be able to manage them. However, to the best of our knowledge, at the moment there is no technique that can automatically annotate the layout components of digital documents, without mapping the document itself to a strict and static template, or using a grammar (in some cases induced by means of a syntactic parsing of the document) representing its layout. On the other hand, writing useful models to understand a particular kind of document can be a demanding task, possibly unfeasible in a weakly structured environment like the Web. This work proposes the use of a concept learning system to infer rules able to identify the document type and its significant components from its layout structure. Specifically, since in digital libraries new and different documents typically become available over time and are to be integrated in the collection, we consider incrementality as a fundamental requirement for the techniques to be adopted. Indeed, it can allow the system to update or revise at any moment a faulty knowledge previously acquired for identifying the logical structure of documents. Furthermore, the inborn complexity of the document domain, in which one cannot know *a priori* how many components make up a generic document and information on the topological structure of the document's components can be crucial for the understanding process, require the ability to express relations among components, and suggest to exploit first-order logic as a powerful symbolic representation language, that also shows human comprehensibility of the resulting learned rules as an additional desirable feature.

In this work we focus on the integration of an incremental learning system in DOMINUS (DOcument Management INtelligent Universal System), a system characterized by the intensive exploitation of intelligent techniques in each step of document processing, from acquisition to indexing,

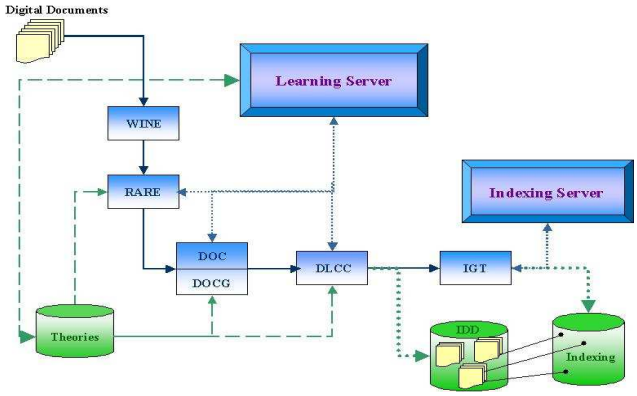

Figure 1. DOMINUS Architecture

categorization, storage and retrieval (technical details can be found in [5]). After a brief description of the architecture of DOMINUS, the learning system is presented along with operation examples in the document management system and an application to a real dataset.

2 DOMINUS: Overall Architecture

The architecture of DOMINUS is depicted in Figure 1. The layout analysis process on documents in digital format starts with the application of a pre-processing module, called WINE (Wrapper for the Interpretation of Non-uniform Electronic document formats), that takes as input a digital document (currently in PS or PDF format) and produces (by an intermediate vector format) its initial XML description as a set of pages made up of basic blocks. Due to the large number of such blocks, that often correspond to fragments of words, it is necessary a first aggregation based on blocks overlapping or adjacency, yielding composite blocks corresponding to whole words. The number of blocks after this step is still large, thus a further aggregation (e.g., of words into lines) is needed. Since grouping techniques based on the mean distance between blocks proved unable to correctly handle the case of multi-column documents, such a task was cast to a multiple instance problem and solved exploiting the kernel-based method proposed in [4], implemented in the Learning Server module. It is able to generate rewriting rules that suggest how to set some parameters in order to group together word blocks to obtain lines. The inferred rules will be stored in the Theories knowledge base for future exploitation and modification by RARE (Rule Aggregation REwriter).

Once the line-block representation is generated, DOC (Document Organization Composer) collects the semantically related blocks into groups by identifying the surrounding frames based on the results of a background structure analysis performed according to an improved version of Breuel's algorithm [2], that finds iteratively the maximal white rectangles in a page: here the process is stopped before finding insignificant white spaces such as inter-word or inter-line ones. At the end of this step, some blocks might not be correctly recognized. In such a case a phase of layout correction is started by DOCG (Document Organization Correction Generator) by exploiting embedded rules stored in the Theories knowledge base. Such rules were automatically learned, using the incremental learning system (see next section) embedded in the Learning Server, from previous manual corrections collected on the initial documents.

Once the layout structure has been correctly and definitely identified, a semantic role must be associated to each significant component in order to perform the automatic extraction of the interesting text with the aim of improving document indexing. This step is performed by DLCC (Document and Layout Components Classifier) by firstly associating the document to a class that expresses its type and then labelling every significant layout component with a tag expressing its role. Both tasks are performed according to theories, previously learned and stored in the Theories knowledge base, that in case of failure can be properly updated. Theory learning and revision are carried out by the first-order incremental learning system, that runs on the new observations and tries to modify the old theories in the knowledge base. At the end of this step both the original document and its XML representation, enriched with class information and components annotation, are stored in the Internal Document Database, IDD. Finally, the text is extracted from the significant components and the Indexing Server is called by the IGT (Index Generator for Text) module to exploit such information for an effective and efficient content-based document storage and retrieval.

3 The Incremental Learning Core

A central role is played by the Learning Server, which intervenes during different processing steps to continuously adapt the knowledge taking into account new evidence and changes in the context. The most important component in such a module is INTHELEX (see [6] for technical details), an incremental learning system able to induce first-order logic theories for multiple concepts/classes starting from a set of examples. Starting from an empty theory, the system considers one by one the available examples, continuously revising the learned theory until it passes the desired accuracy threshold. At that moment, the theory becomes operational and is used to classify new unseen documents. When a reject occurs, a revision is started (although the classification task can still proceed with the old theory until a new one is available) that can even result in the addition of a completely new class learned from just one instance.

INTHELEX incorporates two refinement operators to

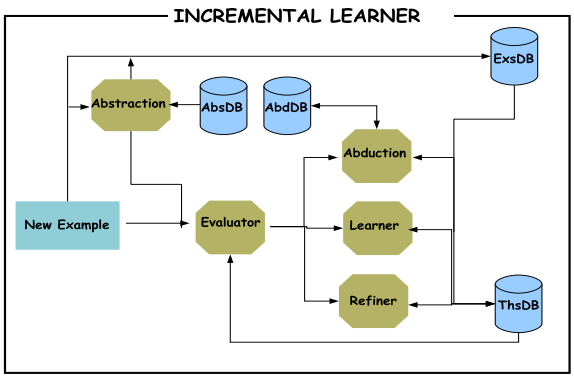

Figure 2. Learning System Architecture

correct the wrong behavior of the current theory: one for generalizing definitions that reject positive examples and the other for specializing definitions that explain negative examples. In the former case, the system can: 1) Drop some conditions from one of the available definitions of the concept the example refers to, so that the resulting revised theory covers the new example and is consistent with all the past negative examples; 2) Add to the current theory (if consistent) a new alternative definition of the concept based on the misclassified example; 3) Add the positive example to the theory as an exception. In the latter case, among the theory definitions that concur in explaining the example, the system can: 1) Add to it some conditions which characterize all the past positive examples and can discriminate them from the current negative one; 2) Add the negation of a condition, that is able to discriminate the negative example from all the past positive ones; 3) Add the negative example to the theory as an exception. New incoming observations are always checked against the exceptions before applying the rules defining the concept they refer to.

Another peculiarity of the system is the exploitation of multistrategy (specifically, Abduction and Abstraction) operators that, if selected, may help in solving the theory revision problem by pre-processing the incoming information. Abduction can hypothesize unseen facts that, together with a given theory, could explain a new incoming observation, this way completing possibly partial information in the examples (adding more details). Abstraction can perform a shift from the language in which examples and the learned theory are described to a higher level one, by means of a set of operators that can replace a number of components by a compound object, or decrease the granularity of a set of values, or ignore whole objects or just part of their features, or neglect the number of occurrences of some kind of object.

The overall learning cycle, according to Figure 2, can be described as follows. A set of examples of the concepts to be learned, possibly selected by an expert, is provided by the environment. Whenever a new example is taken into account, it (or its abstracted version obtained exploiting the *Abstraction* module and database, AbsDB) is stored in the database of processed examples (ExsDB) and checked by the *Evaluator* against the current theory. If the theory is correct the system steps to the next example (if any), otherwise the *Evaluator* tries to avoid the refinement step using the *Abduction* Module and database (AbdDB), to hypothese information that can consistently complete the observation. If this cannot be done then the *Evaluator* can refine the existing theory by means of the *Refiner* (described above) or, in case of an example referring to a completely new class/concept, use the *Learner* (steps 2 and 3 of the generalization phase). Both modules check their result against the examples stored in ExsDB and the theory learned so far.

3.1 Operation Examples

Suppose that the theory learned so far says that: "*a document belongs to class A if, among other blocks, it contains a block (block3) in the top-left part*"[1]:

```
class_A(doc) :-
 part_of(doc,block1),part_of(doc,block2),
 part_of(doc,block3),part_of(doc,block4),
 text(block1),text(block2),
 posupper(doc,block3),posleft(doc,block3),..
```

Suppose that this block represents a date and that a new positive example for class A contains the date on the bottom-left part. To correctly classify such a new example, if the vertical position is not significant for distinguishing the date, the system generalizes the definition by dropping a condition as follows:

```
class_A(doc) :-
  part_of(doc,block1),part_of(doc,block2),
  part_of(doc,block3),part_of(doc,block4),
  text(block1),text(block2),
  posleft(doc,block3),.................
```

saying that "*a document belongs to class A if, among others, there is a block on the left of the document*". Otherwise, if this violates consistency against past negative examples, the theory is revised by adding an alternative definition:

```
class_A(doc) :-
  part_of(doc,block1),part_of(doc,block2),
  part_of(doc,block3),part_of(doc,block4),
  text(block1),text(block2),
  posupper(doc,block3),posleft(doc,block3),
  ....................................
```

[1] A brief description of the predicate symbols exploited in the document representation follows. For each layout component C identified in the document structure, the part_of(D,C) indicates that it is part of a document D; the unary predicate symbols (attributes) describe properties of the layout component (e.g. height, width, type of component (text, graphic, mixed, vertical_line, horizontal_line), position in the page (vertically-upper/middle/lower, horizontally-left/center/rigth)); the n-ary predicate symbols (relations) express relationships between layout components (position of a component with respect to other (vertically-top/middle/bottom, horizontally-left/center/rigth), one is on top/to right the other).

```
class_A(doc) :-
  part_of(doc,block1),part_of(doc,block2),
  part_of(doc,block3),part_of(doc,block4),
  part_of(doc,block5),part_of(doc,block6),
  text(block1),text(block2),
  poslower(doc,block3),posleft(doc,block3),
  ..................................
```

meaning that "*a document belongs to class A if, among other blocks, there is a block in the top-left* or *bottom-left part*". On the other hand, suppose that the new negative example for class A is very similar to the documents analyzed so far (it contains a block on the top-left part), with the only difference of the block type: in the positive examples it represents a date and thus it is *text*, in the negative example it represents a logo and thus it is *graphic*. In such a situation the system specializes the theory by adding a condition:

```
class_A(doc) :-
  part_of(doc,block1),part_of(doc,block2),
  part_of(doc,block3),part_of(doc,block4),
  text(block1),text(block2),
  posupper(doc,block3),posleft(doc,block3),
  text(block3),...................
```

i.e. "*a document belongs to class A if, among other blocks, there is a* text *block in the top-left of the document*".
Before performing the theory revision, the system exploits its multi-strategic capabilities to process the description itself, in order to make computationally easier the revision step, and/or to avoid it completely, if possible. For example, the abstraction operator can *clean* the document description from details uninteresting for learning a useful description for the document class. Suppose that the new document description contains a block (block3) that is an horizontal line which was not present in previous positive examples:

```
class_A(doc) :-
  part_of(doc,block1),part_of(doc,block2),
  part_of(doc,block3),part_of(doc,block4),
  text(block1),text(block2),
  posupper(doc,block3),posleft(doc,block3),
  horizontal_line(block3),.............
```

The abstraction operator can eliminate this block and its features from the description, resulting in:

```
class_A(doc) :-
  part_of(doc,block1),part_of(doc,block2),
  part_of(doc,block4),
  text(block1),text(block2),.............
```

The revision process could be avoided by exploiting the abductive operator to hypothesize, for example, the value of missing features in the document description due to possible noise introduced by the layout analysis step. In case of success, the hypothesized features are stored for future processing; otherwise, the theory revision step is performed. Suppose that the theory learned so far says that "*a document belongs to class A if, among others, there is a text block (block3) on the top-left part of the document*":

```
class_A(doc) :-
 part_of(doc,block1),part_of(doc,block2),
 part_of(doc,block3),part_of(doc,block4),
 text(block1),text(block2),text(block3),
 posupper(doc,block3),posleft(doc,block3),..
```

Suppose that this block represents a date and that a new document description, d1, does not contain such a block due to a failure of the layout analysis step:

```
class_A(d1) :-
 part_of(d1,block1),part_of(d1,block2),
 part_of(d1,block4),
 text(block1),text(block2),..........
```

Since the resulting example could not be recognized as belonging to class A, before revising the theory (which would eliminate a possibly important detail) the system tries to hypothesize the presence of the date completing the new example description with such an information, provided that this is consistent with the theory and previous assumptions:

```
class_A(d1) :-
 part_of(d1,block1),part_of(d1,block2),
 part_of(d1,block3),part_of(d1,block4),
 text(block1),text(block2),text(block3),
 posupper(d1,block3),posleft(d1,block3),
 ...................
abduced[part_of(d1,block3),text(block3),
   posupper(d1,block3),posleft(d1,block3)].
```

The learned theories are represented in a language that can be naturally read by humans. Indeed, the end user of DOMINUS could ask to translate them in natural language.

4 Evaluation

This section presents experimental results of the incremental system on the layout correction, classification and understanding phases of the document processing task. The dataset consisted of 353 documents coming from a collection of publications of the last ten years available online (e.g., in publishers' sites, authors' home pages, scientific paper repositories). Four layout styles (classes) were identified: Springer-Verlag Lecture Notes series (svln, 70 documents), Elsevier journals (61), Machine Learning Journal (mlj, 122 documents: formerly Kluwer Academic, now Springer Science publishers) and Journal of Machine Learning Research (jmlr, 100). Since even documents in the same class might follow different layout standards according to the time of publication and changes in spatial organization of the first page might affect the classification step, the incremental abilities of the learning system are necessary to accommodate the various concept definitions in time.

After the layout analysis step, each document page was automatically described by its number and position (whether it is at the beginning, in the middle or at the end of the document), and by its blocks and frames size, type, position and topological relations (e.g. closeness, intersection, overlapping and inclusion). Each document, resulting in a description made up of an average of 400 predicate instantiations, was considered as a positive example for the class

Table 1. Learning System Performance

MLJ	Rev	Rev+	Rev-	Acc.
Title	59.6 (12%)	38.2 (35%)	21.4 (6%)	91.1%
Abstract	50.7 (11%)	39.6 (38%)	11.1 (3%)	92%
Author	52.1 (11%)	39.7 (25%)	11.4 (4%)	94.2%
Keywords	34.2 (7%)	27.9 (26%)	6.3 (2%)	97%
Elsevier	Rev	Rev+	Rev-	Acc.
Title	18.2 (7%)	13.6 (24%)	4.6 (2%)	95.5%
Abstract	28.9 (11%)	22.1 (4%)	6.8 (3%)	96.6%
Author	21.4 (8%)	16.6 (31%)	4.8 (2%)	95.2%
Keywords	17.2 (7%)	15.5 (36%)	1.7 (1%)	96.2%
Logo	5.8 (2%)	5.8 (11%)	0 (0%)	99.6%

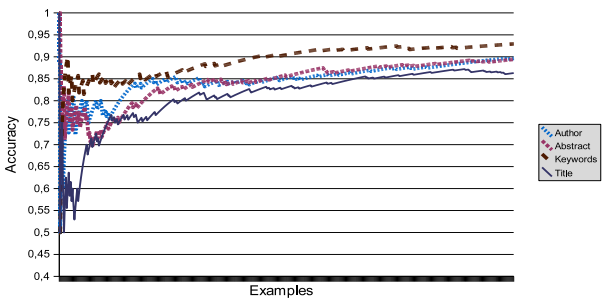

Figure 3. Accuracy on tuning phase

it belongs to, and as a negative example for all the other classes to be learned. The system performance was evaluated according to a 10-fold cross validation methodology.

The first experiment was run to infer rules for layout correction: 1897 manual corrections were collected, described (representing the situation before and after the correction) and provided to the learner, that inferred correction rules showing a 97.7% average accuracy. The second experiment concerned the induction of classification rules, and obtained good performance in terms of runtime (52 sec.-elsevier, 399 sec.-svln, 588 sec.-jmlr, 3213 sec.-mlj), accuracy (100%-elsevier, 97.73%-svln, 98.3%-jmlr, 95.75%-mlj) and number of theory revisions (6.3-elsevier, 11.8-svln, 26.4-mlj, 11.4-jmlr) on an average of 320 examples in each fold. Here we report the best (elsevier) and worst (mlj) performing classes on document image understanding. Although we focus on first page layout components, it should be noted that DOMINUS is able to handle multi-page documents, and hence could also recognize components placed in other pages, such as bibliographic references, that can be a valuable source of information.

Table 1 reports the experimental results, averaged on the 10 folds. Predictive accuracy (Acc) is always very high. Furthermore, the number of revisions performed, both in absolute value (Rev = overall, Rev+ (Rev-) = on positive (negative) examples only) and in ratio (in parentheses), confirm the cautious behavior of the system in learning definitions (and thus a model for the classes/layout components) that preserve as much information as possible from the original training examples, so that generalizations are needed more often than specializations. Figure 3 shows the accuracy evolution during the incremental induction of layout components definitions on mlj class: after a first phase in which many revisions have to be performed to restore the theory correctness, resulting in a poor accuracy, it tends to increase towards a stable condition meaning that the system was able to grasp the correct layout component definitions.

5 Conclusion

The huge amount of documents available in digital format and the flourishing of digital repositories raise problems concerning effective and efficient document storage and retrieval. This paper highlights how systems for document management can take advantage by the intensive application of Machine Learning, and specifically of incremental first-order logic learning techniques. Indeed, experiments on a real-world dataset have been presented and discussed, that confirm the validity of the proposed approach.

References

[1] A. Anjewierden. AIDAS: Incremental logical structure discovery in pdf documents. In *Proc. of ICDAR 2001*, pages 374–378, 2001.

[2] T. M. Breuel. Two geometric algorithms for layout analysis. In *Workshop on Document Analysis Systems*, 2002.

[3] H. Chao and J. Fan. Layout and content extraction for pdf documents. In *Proc. of DAS 2004*, volume 3163 of *LNCS*, pages 213–224, 2004.

[4] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[5] F. Esposito, S. Ferilli, T. Basile, and N. D. Mauro. Automatic content-based indexing of digital documents through intelligent processing techniques. In *Proc. of DIAL 2006*, pages 204–219, 2006.

[6] F. Esposito, S. Ferilli, N. Fanizzi, T. M. Basile, and N. Di Mauro. Incremental multistrategy learning for document processing. *Applied Artificial Intelligence: An Internationa Journal*, 17(8/9):859–883, 2003.

[7] R. P. Futrelle, M. Shao, C. Cieslik, and A. E. Grimes. Extraction, layout analysis and classification of diagrams in PDF documents. In *ICDAR 2003*, pages 1007–1014, 2003.

[8] W. S. Lovegrove and D. F. Brailsford. Document analysis of PDF files: methods, results and implications. *Electronic Publishing – Origination, Dissemination and Design*, 8(2-3):207–220, 1995.

[9] M. Rigamonti, J.-L. Bloechle, K. Hadjar, D. Lalanne, and R. Ingold. Towards a canonical and structured representation of PDF documents through reverse engineering. In *Proc. of ICDAR 2005*, pages 1050–1055. IEEE Comp. Society, 2005.