A DTD Extension for Document Structure Recognition *

Rolf Brugger, Frédéric Bapst, and Rolf Ingold

Informatics Institute of the University of Fribourg, Chemin du Musée 3, CH-1700 Fribourg, Switzerland

Abstract. This paper deals with the representation of document models used in the field of document recognition. A novel formalism called *generalized n-gram* is presented, which is shown to be accurate for the recognition task and well adapted to automatic learning by examples. The paper addresses also the thorny problem of integrating models for document analysis with existing standards used for document manipulation and production.

1 Introduction

The benefits of using high level descriptions to handle structured documents has been widely recognized by the scientific community dealing with electronic document production. The SGML language providing the DTD mechanism [11] has become increasingly important during the last five years. The document analysis and recognition community has been dealing with structure recognition for several years too [4, 7, 1]. In this context, the use of general knowledge, often referred to as document models, describing the document class to be processed is essential; it is clear that these document models share many common features with DTDs, at least the generic logical structure.

Very little work has been done so far in order to combine both kinds of formalism [13, 10]. This paper aims to make a significant contribution to reduce the gap between document models used for recognition purposes and document type definitions commonly used for structured document handling. In order to achieve that goal, it is necessary to have a closer look at the features of both formalisms. Statistical information appears to be very useful in pattern recognition in general [8] and in document structure analysis in particular. In the case of documents, for instance, it may be very useful to know the a priori probability of some elements, or more generally the probabilities of combinations of several of them. In this paper, we present a formalism called generalized n-grams, that allows us to represent such statistical knowledge and which is specially well adapted for incremental learning by examples [3].

Section 2 presents the generalized n-grams. Section 3 and 4 focus on its use for the learning and recognition tasks respectively. Section 5 discusses results of experiments we have made in order to validate the approach. Finally, section 6 discusses the problem of integrating the formalism with SGML.

* This project is funded by the Swiss National Fund for Scientific Research, code 21-42'355.94.

2 The Generalized n-Gram Model

A *generic* document model describes the structuring rules of a document class. It defines in a generic way how specific documents are structured. We distinguish two types of generic structures: the generic logical structure which reflects the logical organization of a class of documents and the generic physical structure which represents its layout. A priori, there exist many ways to represent the generic structure of documents. Among the existing formats (SGML, ODA, Thot) we will focus on SGML, first because it is an ISO standard, and second because a multitude of commercial and public domain tools are available for it.

There are two relevant formalisms to express document models: an SGML DTD defines the logical structure and DSSSL [12] defines how logical documents are translated to physical documents. These formalisms were defined to standardize the production of documents. One of their main drawbacks is that they cannot be directly used for the recognition of documents. Therefore, we have developed a model which has been designed to meet the requirements of document recognition. These requirements are: high accuracy, error tolerance and automatic inference. The model is presented in the next section.

2.1 A Statistical Document Model

In our approach, the tree structure of logical documents is represented by probabilities of local tree node patterns similar to n-grams. N-grams are a commonly used natural language model [5], which takes the assumption that only the previous $n-1$ words in a sentence have any effect on the probabilities for the next word. Usually, the n-gram model is used to represent linear structures such as natural language words, etc. In order to apply it to tree structures, it has to be generalized. We chose to consider not only "horizontal" n-grams for siblings but also "vertical" n-grams to take into account ancestor and child relations.

The power of n-grams lies in the fact that they can predict entities in an incomplete logical tree, when their context is known. Additionally, each prediction is weighted with the probability of the corresponding n-gram. The main problem, when n-grams are extended to more complex structures, is to extend the representation of their context accordingly. In the next section we propose a generalization of n-gram models to tree structures.

2.2 Model Specification

In the proposed statistical document model, we represent the context of a node by its local neighborhood, consisting of left and right siblings, the ancestor as well as the first and last descendants. A priori, in our generalization of n-grams for trees any value for $n > 1$ is possible. However, our definitions and tests will always be based on trigrams ($n = 3$). The choice of $n = 3$ is also commonly used in natural language processing [5].

The set S of all logical labels is defined as

$$S = \{S_0, S_1, \ldots, S_m\}$$

where m is the number of logical labels. The special label S_0 indicates a "non-existent node" and is used when a node has no direct neighbor in a specific direction.

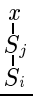
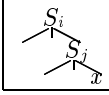
Tree pattern	Relation between x and S_i, S_j	Notation
	The node x is the left neighbor of the pattern $S_j\ S_i$.	$L(x, S_i, S_j)$
	The node x is the right neighbor of the pattern $S_i\ S_j$.	$R(x, S_i, S_j)$
	The node x is the ancestor of S_j, which is the ancestor of S_i.	$A(x, S_i, S_j)$
	The node x is the first descendant of S_j which is the first descendant of S_i.	$B(x, S_i, S_j)$
	The node x is the last descendant of S_j which is the last descendant of S_i.	$E(x, S_i, S_j)$

Table 1. Notation used to represent tree patterns.

In the following, the relations L, R, A, B and E are used respectively to express left sibling, right sibling, ancestor, first (begin) child and last (end) child patterns in a tree (refer to Table 1). Supposing that the nodes in a tree corresponding to the logical entities $S_1, \ldots, S_m$ are labeled, we can define the trigram

$$l_{i,j,k} = P(x = S_k \mid L(x, S_i, S_j))$$

as the probability that the node S_k is the left sibling of the node sequence S_j—S_i. The trigrams for the other directions are similarly defined:

$$\begin{aligned} r_{i,j,k} &= P(x = S_k \mid R(x, S_i, S_j)) \\ a_{i,j,k} &= P(x = S_k \mid A(x, S_i, S_j)) \\ b_{i,j,k} &= P(x = S_k \mid B(x, S_i, S_j)) \\ e_{i,j,k} &= P(x = S_k \mid E(x, S_i, S_j)). \end{aligned}$$

These basic trigrams enable us to define an entire document model. The complete model M of a document class is the tuple of all logical entities S and the set of all trigrams G in a document class:

$$\begin{aligned} M &= (S, G) \\ G &= \{l_{\alpha,\beta,\gamma}, r_{\alpha,\beta,\gamma}, a_{\alpha,\beta,\gamma}, b_{\alpha,\beta,\gamma}, e_{\alpha,\beta,\gamma}\} \\ &\quad with\ 0 \leq \alpha \leq m, 1 \leq \beta, \gamma \leq m \end{aligned}$$

We will now describe how the proposed document model can be generated from a set of example documents.

3 Learning the Document Model

The learning algorithm must be able to construct the entire model from some examples, which are represented with their logical structure trees or subtrees. All trees are assumed to belong to the same document class.

These trees can for example be generated interactively by the user. After having integrated some samples, the system should be able to run the recognition on a new document. If the recognition (partially) fails, corrections are manually performed and the corrected tree is passed to the learning algorithm in order to update the model. This incremental learning procedure is an important feature of our approach.

To integrate a new example into the description of the left and right siblings, first and last descendants and ancestor directions we use the same algorithm for every new example. The example tree T_{ex} is traversed and all trigrams are extracted. The probability

$$P(x = S_k \mid L(x, S_i, S_j)) = \frac{P(L(S_i, S_j, S_k))}{P(L(S_i, S_j))}$$

can therefore be estimated as:

$$\hat{P}(x = S_k \mid L(x, S_i, S_j)) = \frac{F(L(S_i, S_j, S_k), T_{ex})}{F(L(S_i, S_j), T_{ex})}$$

where $F(L(S_i, S_j, S_k), T_{ex})$ and $F(L(S_i, S_j), T_{ex})$ indicate the frequencies the patterns $L(S_i, S_j, S_k)$ and $L(S_i, S_j)$ were found in T_{ex}. $L(S_i, S_j)$ specifies a pattern of two entities, where S_i is the left neighbor of S_j. The conditional probabilities for the right, ancestor, begin and end contexts are estimated similarly, just by replacing the L relations by R, A, B or E relations.

Here are some comments on the statistical part of the learning method:

- Incremental learning is trivial, since only estimated probabilities have to be updated.
- The learning method is independent of the order in which learning samples are provided to the system.
- It is not necessary to keep a tree sample after it has been integrated into the model. This is different to a rule based approach, where it is often necessary to store samples after the rules have been generated to enable incremental learning.
- There are no restrictions on the size or "shape" of a sample. It can be a tree of an entire document or just a small part of it.

4 Recognition of the Logical Document Structure

Our recognition problem belongs to the class of structural pattern recognition. The input is a logical document model and a sequence of elementary physical entities. The sequence corresponds to the leaves of a specific physical structure tree.

The aim of the recognition task is to construct a logical tree on top of the physical entities in conformity to the model. As there may be several solutions that conform to the model, the most likely one has to be found. This is done by means of a conformity measure between the model and a specific tree.

4.1 A Heuristic Conformity Measure

In the optimization process we need a measure which computes the conformity of a tree (global) or subtree (local) with a model. The measure should be able to meet the following requirements:

- The conformity (coherence of a subtree) should increase if nodes are aggregated in conformity with the model and decrease if the (partial) tree is incoherent. A tree is incoherent if it contains patterns whose corresponding trigram probabilities are zero.
- In order to achieve error tolerance, the conformity should not be decreased much by singular incoherent patterns.
- The overall conformity of an entire tree is determined by the local quality measures of its subtrees. It might be optimal even if some of the subtrees aren't.

The model is used to locally evaluate trigram patterns within the considered tree. For instance, the conditional trigram probability can be interpreted as a plausibility for each trigram that was found in a tree. This plausibility ranges from 0 to 1, where 0 means "totally impossible pattern with respect to a model" and 1 "given a specific neighborhood, the existence of an element is certain". These trigram plausibilities of a tree are collected and integrated into a plausibility value for the entire tree. An integration formula, which is used in the area of processing of uncertain information, is the generalized means function [9]. The conformity function is then expressed as:

$$q(a_1, \ldots, a_n, \alpha) = \left(\frac{p(a_1)^\alpha + p(a_2)^\alpha + \ldots + p(a_n)^\alpha}{n} \right)^{\frac{1}{\alpha}}$$

where n represents the number of patterns in the tree and the $p(a_i)$ represent the trigram plausibilities $l_{i,j,k}$, $r_{i,j,k}$, etc. of the corresponding patterns a_i. As shown in Figure 1, the generalized means function can seamlessly be tuned between the standard fuzzy OR and fuzzy AND functions by varying the parameter α between $-\infty$ and $+\infty$.

The parameter α of the integration function is best chosen to be somewhere between ($\alpha \in [-\infty, 1]$). Our assumption is that an entire tree is coherent if all or most of its trigrams are quite plausible.

	fuzzy AND	harmonic mean	geometric mean	arithmetic mean	fuzzy OR
$\alpha =$	$-\infty$	-1	0	$+1$	$+\infty$

Fig. 1. Functions obtained by varying the parameter α of the generalized means function.

4.2 Construction of Logical Trees

The recognition consists of building the optimal logical tree on top of the physical entities. A priori, there exists an exponential number of trees that can be built over

a given sequence of leaves. In order to reduce the time spent in finding an optimal solution, heuristics must be applied. For our prototype, we have chosen a *best-first* [14] search algorithm — a classical, well understood and easy to implement optimization method.

The principle of the construction of a logical tree is that first, the physical entities are considered as the leaves of the logical tree; second, the logical tree is built on them step by step by adding nodes. In a bottom-up manner, new nodes are used to group together leaves and other previous nodes. According to the best-first algorithm, many competing trees might exist at a time. The current best one, according to the conformity measure, is selected for further node aggregations. The recognition terminates, when a tree including all leaves is found.

As an illustration, let us consider the following scenario: The recognition starts with the leaves in the logical tree to be searched. The leaves are considered as an incomplete or partial solution p_0. The system then makes a hypothesis for a sequence of leaves to be grouped together to build a subtree. The obtained new partial solution p_1 is then tested for its coherence, which is based on the conformity values of all subtrees of p_1. If the conformity value of p_1 is higher than that of p_0, a new hypothesis for grouping together nodes into a subtree is created to obtain the next partial solution p_2. Otherwise, the node that has been added to p_0 is a wrong one or it might have grouped together a wrong set of nodes. In this case, the system restarts with p_0 but tries another hypothesis about the node to add, hoping for a better guess.

In order to quickly obtain meaningful logical trees, the aggregation of nodes is guided by the document model. The model with its trigram information permits the system to make precise hypotheses on which new nodes are most probable.

Notice that the recognition method is inherently error tolerant. If a physical entity was badly recognized so that it is not possible to construct an entirely coherent tree, there is still a good chance of finding a nearly correct tree. It is an important advantage of the approach that the system not only tries to find as many parts as possible of the correct tree but also to recognize incorrect parts in order to reject them. These incorrect parts can easily be found: they are the local parts, where the local tree incoherence could not be eliminated, even by trying all possible variants. In such a case, either the model has to be extended with this sample, or the physical information needs to be revised.

5 Results

Due to their complexity, the statistical document model and the learning and recognition algorithms are difficult to discuss on a theoretical level. The approach has to be properly evaluated on “real world data” through a deep analysis of (i) the power of the proposed model to guide document recognition and (ii) the power of the incremental learning in the generation of stable models.

We have developed a prototype for a first evaluation of the approach. It has been applied to two sets of documents: scientific articles and multicolumn list of academic activities at our university. Here, we will be summarizing the recognition results for the determination of the macro structure of scientific articles. The macro structure consists of the logical tree structure from the root down to the line nodes of a document. The test

set consists of 20 pages drawn from six articles containing some 1600 logical nodes. The articles use a non trivial structure including chapters, subchapters, enumerations, bibliographical references, mathematical formulas, etc.

Figure 2 summarizes the tests. It shows how many logical nodes and which documents were used to learn the model, which document was recognized and the number of incorrectly recognized nodes. The error rate in parentheses designates the number of errors in the solution considered to be the second best according to the quality measure. We can see that the second best solution in reality corresponds to a better solution than the one with the higher score. This can be explained with a non optimal definition of the parameters of the quality measure function. Better choices of these parameters are subject to future evaluations.

Learning		Recognition		Errors	
#nodes	documents	#nodes	document	absolute	relative
630	3,4,5,6	859	1	3 (0)	0.3%
629	2,4,5,6			3 (0)	0.3%
470	2,3,4			3 (0)	0.3%
376	2,6			5	0.5%
629	2,4,5,6	170	3	4 (2)	2.4%
435	2,4,5			5 (6)	2.9%
376	2,6			5 (6)	2.9%
630	3,4,5,6	171	2	model incomplete	-
470	2,3,4	133	5	3 (0)	2.2%
470	2,3,4	195	6	4 (1)	2.0%

Table 2. Test results for the recognition of the macro structure of scientific articles.

The Table shows that the documents were recognized with few errors in general. It also shows the improved recognition rates when incremental learning is applied. If only the documents 2 and 6 were learned, the documents 1 and 3 were both recognized with 5 wrong nodes each. Incrementally adding the documents 4 and 5 reduces the number of wrong nodes to 3 and 4.

Document 2 could never be recognized because it is the only one which contains a specific physical element that wasn't part of the learning set. An element can't be recognized if it has never been learned beforehand. Referring finally to the last line of Figure 2, the experiments also showed that three carefully selected documents can be sufficient to establish a complete model.

Given these results we can claim from our experiment that the use of representative samples and incremental learning lead to a model converging towards a stable state.

6 Interfacing with SGML

Most applications dealing with structured documents use DTDs written in SGML to describe the generic logical structure of documents and DSSSL to specify their pro-

cessing; in particular the DSSSL style language enables the specification of the typographic style of documents. The generalized n-grams formalism described so far has been shown adequate for document structure learning and recognition. We now have to examine how it can be combined with the SGML world.

The information contained implicitly in the generalized n-gram model covers in fact three different parts:

- a generic description of structures;
- a typographic style description;
- statistical information on frequencies of structural patterns.

The first part may be compared to the generic logical structure as specified by DTDs in SGML; however transformations are necessary to match the SGML model. Section 6.1 presents a method of translating the generalized n-gram paradigm to element definitions in SGML. The second part dealing with typographic features can be handled in a natural way, as described in Section 6.2. Our study has not developed this subject in detail.

The statistical information handling appears to be more delicate, since no equivalent exists for document manipulation and production. Conceptually this information should be added to the generic structure description, but there is no mechanism for specifying this kind of information in a DTD. We will outline a solution to this problem in Section 6.3.

6.1 Inference of Grammatical Rules

The experiments described in the previous chapter proved that the model is expressive enough to enable successful recognition of document structures. We wanted to know whether the model is even expressive enough to enable the inference of grammatical rules, which describe the structure of the learned documents. Such rules could then easily be translated to an SGML-DTD, which opens the world of SGML for the statistical model.

To test this, we have developed a prototype for rule inference that is exclusively based on the generalized n-gram model [2]. We chose the following approach for rule inference. First, an initial DTD is created which will be far too general. For this, only the n-grams describing node-ancestor relations are used. Second, these initial rules are iteratively specialized using node-sibling, node-first-child and node-last-child information.

To illustrate the inference of rules, let us consider the following example. Assume that the system has been presented some subchapter structures which consist of a subchapter node (`SChapter`) whose children nodes are a subchapter title (`ScTitle`) followed by an undefined number of paragraphs (`Parag`). An example of a subchapter tree is shown in Figure 2.

After having learned some subchapter trees, the statistical model might look like shown in the Table 3. For simplicity, only bigrams and a small subset of the model are shown.

The inference of rules can start, if a complete model was learned. This means that all relevant n-gram probabilities should be available. The principle of the inference is first

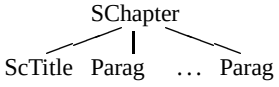


Fig. 2. Subchapter structures.

Neighborhood of ScTitle	left Neighborhood of Parag	right Neighborhood of Parag
$a_{\mathrm{SChapter,ScTitle}} > 0$	$a_{\mathrm{SChapter,Parag}} > 0$	$r_{\mathrm{S_0,Parag}} = 0.1$
$l_{\mathrm{S_0,ScTitle}} = 1$	$l_{\mathrm{Parag,Parag}} = 0.9$	$r_{\mathrm{Parag,Parag}} = 0.9$
$r_{\mathrm{Parag,ScTitle}} = 1$	$l_{\mathrm{ScTitle,Parag}} = 0.1$	$r_{\mathrm{x,Parag}} = 0$, for
$l_{\mathrm{x,ScTitle}} = 0$, for	$l_{\mathrm{x,Parag}} = 0$, for	all other x.
all other x.	all other x.	

Table 3. Extract of the bigram model for subchapter structures.

to create a trivial, initial set of rules and then to specialize these rules step by step using the information provided by the n-grams. This finally leads to a set of more expressive rules. Here is an example for the inference of the subchapter rule:

1. First all entities which are potential children of a subchapter node (`SChapter`) are collected. They are the nodes x, whose bigram probabilities $a_{\mathtt{SChapter,x}}$ is greater than zero ($a_{\mathtt{SChapter,ScTitle}}$, $a_{\mathtt{SChapter,Parag}}$). Using these bigrams, the following initial rule can be found:

$$\mathtt{SChapter} \Rightarrow (\mathtt{Parag} \mid \mathtt{ScTitle})^*$$

2. Due to the fact that a `Parag` is never followed by a `ScTitle` ($r_{\mathtt{ScTitle,Parag}} = 0$), that `ScTitle` can be the first node in a subtree ($l_{\mathtt{S_0,ScTitle}} > 0$), and that `Parag` can be the last node in a subtree ($r_{\mathtt{S_0,Parag}} > 0$), the initial rule can be specialized to:

$$\mathtt{SChapter} \Rightarrow \mathtt{ScTitle}^*\ \mathtt{Parag}^*$$

3. Finally, a `ScTitle` can never be followed by another `ScTitle` ($l_{\mathtt{ScTitle,ScTitle}} = 0$), but an entity `Parag` can be followed by another entity `Parag` ($r_{\mathtt{Parag,Parag}} > 0$). Therefore the rule can be further specialized to:

$$\mathtt{SChapter} \Rightarrow \mathtt{ScTitle}\ \mathtt{Parag}^*$$

The prototype outputs a standard SGML-DTD which can therefore easily be compared to human generated DTDs. The results were very encouraging: with a complete trigram model the generated rules were accurate and pertinent. The automatically generated rules basically do not differ from the human generated ones. Slight differences were found, for example, when a threshold is used to infer an iteration from a sequence of identical entities. Thus the results of the experiment have shown that the trigram model for trees integrates relevant information adequate to describe document classes.

6.2 Typographical Attributes

The DTD formalism has been designed to offer a flexible way for the representation of logical document structures. The typographical representation of these documents can only be integrated into a DTD with difficulty — there is no standard for it. This situation has been remedied by the definition of the DSSSL standard.

In the context of the learning of document structures an ideal system should infer a DTD as well as the appropriate DSSSL description for a set of documents. Both could then, for instance, be used to regenerate the initial physical documents (possibly after having edited the content or presentation of the documents). The main problem with this is the complexity of the DSSSL formalism. In fact, DSSSL is an offspring of the LISP family and therefore a complete, fully-fledged programming language. The inference of a DSSSL program based on some document examples is a task that needs human intelligence. It lies far beyond the possibilities of current inference systems.

However, the problem can be eased by limiting the inference to a subset of DSSSL. Many simple DSSSL descriptions are mainly composed of uniquely declarative parts as shown in Figure 3. The inference of such declarations can be achieved by a simple system which calculates the maximum likelihood of the predefined typographical parameters like `font-size` and associates them with the relevant logical entity `Parag`.

```
(element Parag
    (make paragraph
        font-family-name: "Times"
        font-size:        12pt
        qadding:          'left
    )
)
```

Fig. 3. Example of a DSSSL declaration.

The inverse transformation of a DSSSL description back to the typographical attributes of the statistical model is more complex. In order to be compatible to the standard, it should be possible to process any DSSSL specification to create an initial model. This can only be done by completely interpreting a given DSSSL specification. However, there is no need to re-implement an entire DSSSL engine. Jade [6], one of the most popular public domain DSSSL processors, provides different output formats calculated by different backends. Thus, what we need is a backend processor which is capable of updating the typographical information in the statistical model.

6.3 Statistical Information

Neither a DTD nor DSSSL nor any other formalism of the SGML family offers the possibility of adding statistical information to a document model. In order to obtain an equivalence between the n-gram model and the SGML formalisms the latter have to be extended.

A trivial possibility is to simply copy the n-gram probabilities to a SGML structure for which a DTD has been defined. There are two main drawbacks of this approach: first, the n-gram probabilities represent very low level information and are virtually unreadable by humans. A higher level representation would be useful. Second, in Section 6.1 we showed that a DTD can be inferred out of the set of n-grams. All the information which is contained in a DTD is therefore already contained in the n-grams. This redundancy should be avoided.

A more optimal approach is to associate attributes to previously inferred DTD rules. These attributes represent the number of applications of the DTD rules that would have been necessary to generate the documents of the learning set. In most cases these frequencies, in conjunction with the DTD rules, are sufficient to regenerate a complete n-gram model. The only potential ambiguity arises when the same logical element appears more than once at the right hand-side of the DTD rules. In these cases, it is not clear how many times the element was generated in specific contexts. Figure 4 outlines how the context information could be added. It simply specifies how often the element `Parag` was generated in a chapter or in a subchapter (`SChapter`).

Notice also, that the formalism as proposed here expresses the statistical properties at a very high abstraction level. This even enables humans to read or edit such additional information files. However, we have only presented a preliminary version of the formalism which has to be consolidated in the near future.

```
Article:  3
Title:    3
Chapter:  7
Parag:   27
  Parag(Chapter):   3
  Parag(SChapter): 24
...
```

Fig. 4. Frequency attributes associated to DTD rules.

7 Conclusion

This paper has dealt with document models used in the field of document analysis and recognition. A new generic model based on generalized n-grams has been presented: it has been shown to be accurate enough to recognize logical structures and suitable for easy incremental learning.

The problem of integration of this model with the SGML/DSSSL world has also been addressed. In fact, it has been shown that the generalized n-grams model can be used to generate DTDs in SGML and document style definitions in DSSSL automatically. The problem of translating the statistical knowledge remains partially open and needs further investigations.

We are convinced that the use of common tools for document recognition and document production can notably increase the capacity of both domains. First DTDs and style descriptions bring necessary knowledge to the recognition process; second document models learned from examples and translated to SGML may reduce the cumbersome task of defining DTDs and DSSSL descriptions by hand.

References

1. O. Akindele. *Vers un système de construction automatique de modèles génériques de documents.* PhD thesis, CRIN-Nancy, 1995.
2. D. Bollinger. Inferenz und Spezialisierung kontextfreier Regeln mit statistischen Zusatzinformationen. master's thesis report in computer science, Uni Fribourg, 1996.
3. R. Brugger, A. Zramdini, and R. Ingold. Modeling documents for structure recognition using generalized n-grams. In *ICDAR*, 1997.
4. H. Bunke and P. S. P. Wang. *Handbook of Optical Character Recognition and Document Analysis*. World Scientific Publishing Company, 1997.
5. E. Charniak. *Statistical language learning*. MIT Press, 1993.
6. J. Clark. Jade - james' dsssl engine. http://www.jclark.com/jade/, 1997.
7. P. Fankhauser and Y. Xu. Markitup! an incremental approach to document structure recognition. *Electronic Publishing*, 6, December 1993.
8. D. J. Hand. *Artificial Intelligence, Frontiers in Statistics*. Chapman & Hall, 1993.
9. G. J. Klir and T. A. Folger. *Fuzzy Sets, Uncertainty, and Information.* Prentice-Hall International, 1992.
10. P. Lefèvre and F. Reynaud. ODIL: an SGML description language of the layout structure of documents. In *ICDAR*, 1995.
11. International Standards Organization. Information processing — text and office systems — standard generalized markup language (SGML) (ISO 8879). Geneva: ISO, 1986.
12. International Standards Organization. Document style semantics and specification language (DSSSL) (ISO 10179). Geneva: ISO, 1996.
13. A. L. Spitz. Style directed document recognition. In *ICDAR*, pages 611–619, 1991.
14. P. H. Winston. *Artificial Intelligence*. Addison-Wesley, second edition, 1984.