A knowledge–based architecture for processing documents of a multi–class domain

Enrico Francesconi

Febbraio 1998

Coordinatore: Prof. Ing. Edoardo Mosca
Tutore: Prof. Giovanni Soda
Acknowledgments

I would like to thank Prof. Giovanni Soda, Prof. Marco Gori and Prof. F. Cesarini for their support and guidance during my Ph.D. studies, and all my friends of the Artificial Intelligence laboratory of the University of Florence, for their company and help.
Contents

1 Introduction 1

2 The document automatic reading problem 5
  2.1 Generalities .......................................................... 5
  2.2 Domains and classes of documents ................................. 8
  2.3 Types of architectures for document processing ................. 12
    2.3.1 Closed architectures ........................................... 15
    2.3.2 Open architectures ............................................. 21
    2.3.3 Semi-open architectures ....................................... 26

3 A multi-class document processing architecture 35
  3.1 Introduction .......................................................... 35
  3.2 The semi-open architecture ...................................... 37
    3.2.1 The Document Model ......................................... 38
    3.2.2 The structure of the system ................................ 39
    3.2.3 The Conceptual Model ....................................... 43

4 A Conceptual Model based on a semantic network 47
  4.1 The syntax and the semantics of the network .................... 47
  4.2 A CM for a multi-class domain of documents .................... 50
  4.3 The pragmatics of the network ................................ 53
    4.3.1 The Reading Plan ............................................. 54
    4.3.2 The Document processing procedure ........................ 55
    4.3.3 The extraction of primitives: connected components ....... 60
    4.3.4 Segment extraction and organization ....................... 61
    4.3.5 Segment item rectangles extraction ....................... 62
    4.3.6 Component rectangle extraction ............................ 65
    4.3.7 The Model Driven Labelling procedure ...................... 67
    4.3.8 Keyword based labelling ................................... 77
# List of Figures

2.1 The hierarchical layout structure of a document. 6
2.2 The hierarchical logical structure of a document. 7
2.3 The two schemes of document processing. 9
2.4 The universe of documents divided into domains. 10
2.5 Two examples of documents belonging to different physical classes but to the same logical one. 11
2.6 The relation between a logical class and its physical sub-classes. 12
2.7 A domain of documents divided into classes. 13
2.8 Closed and Open-architecture schemes. 14
2.9 A Semi-open architecture scheme. 15
2.10 Objects on a GIRO check form. 16
2.11 The model graph used of a GIRO check forms. 17
2.12 A document and its tree representation. 18
2.13 An example of a region taxonomy which is used for functional components analysis. 23
2.14 An example of dependence hierarchy among concepts. 28
2.15 The hierarchy of dependencies among concepts. 28

3.1 Example of a document and the related *Document Model*. 39
3.2 The types of physical objects to be extracted. 40
3.3 Flow chart of the system. 42
3.4 The parts of the *Conceptual Model* and the interaction scheme among them. 45

4.1 General data types in the syntactic definition of the network structure. 48
4.2 A general frame for a concept of the semantic network. 49
4.3 A frame description of the concept *DOC_OF_DOMAIN*. 51
4.4 A frame description of the concept *LAY_STRUCT*. 52
4.5 The frame description for the concept *DOC_CLASS_p*. 57
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6</td>
<td>The frame description for the concept $OBJ^P$.</td>
<td>58</td>
</tr>
<tr>
<td>4.7</td>
<td>The recursive procedure to follow an 8-path.</td>
<td>61</td>
</tr>
<tr>
<td>4.8</td>
<td>Segment collections structured in <em>Interval List</em> and in <em>Interval Tree</em></td>
<td>63</td>
</tr>
<tr>
<td>4.9</td>
<td>Intersection on X-axis of the projections of two horizontal segments</td>
<td>64</td>
</tr>
<tr>
<td>4.10</td>
<td>Allen’s 13 interval relations</td>
<td>69</td>
</tr>
<tr>
<td>4.11</td>
<td>The frame description for the concept $Cpt_i$ at the “named object” level.</td>
<td>70</td>
</tr>
<tr>
<td>4.12</td>
<td>An array of the lists of item rectangles related to the states for a</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>particular concept.</td>
<td></td>
</tr>
<tr>
<td>4.13</td>
<td>The scheme of the Model Driven Labelling procedure.</td>
<td>74</td>
</tr>
<tr>
<td>4.14</td>
<td>The generic layout of the semantic network for a multi-class domain of</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>documents.</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Example of invoice and related Document Model. In such a model</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>the object “description” and the related coordinates $(x_{min}, x_{max},$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$y_{min}, y_{max})$ are reported.</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Construction and use of a Document Model for invoice processing.</td>
<td>86</td>
</tr>
<tr>
<td>5.3</td>
<td>A frame description of the concept <em>INVOICE</em>.</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>A frame description of the concept <em>MID_SECT</em>.</td>
<td>88</td>
</tr>
<tr>
<td>5.5</td>
<td>The frame for the concept <em>TOTAL</em>.</td>
<td>89</td>
</tr>
<tr>
<td>5.6</td>
<td>The frame for the concept <em>ITEM_RECT_M</em>.</td>
<td>91</td>
</tr>
<tr>
<td>5.7</td>
<td>The frame for the concept <em>IT_RC_M_wcat(1)</em>.</td>
<td>92</td>
</tr>
<tr>
<td>5.8</td>
<td>Vertical dependence relationships among rectangles</td>
<td>95</td>
</tr>
<tr>
<td>5.9</td>
<td>The frame of the concept <em>QUANTITY</em>.</td>
<td>97</td>
</tr>
<tr>
<td>5.10</td>
<td>Scheme of item rectangle labelling procedure</td>
<td>100</td>
</tr>
<tr>
<td>5.11</td>
<td>The semantic network that describes invoice Conceptual Model.</td>
<td>101</td>
</tr>
<tr>
<td>6.1</td>
<td>Acquisition phase for database creation</td>
<td>106</td>
</tr>
<tr>
<td>6.2</td>
<td>Connected component labelling phase for database creation.</td>
<td>108</td>
</tr>
<tr>
<td>6.3</td>
<td>An original character image and its modification by overlapping</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>some components of synthetic noise.</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>Scheme of character recognition using autoassociators only.</td>
<td>113</td>
</tr>
<tr>
<td>6.5</td>
<td>Scheme of character recognition using the Championship Algorithm.</td>
<td>117</td>
</tr>
<tr>
<td>6.6</td>
<td>Scheme of character recognition using an MLP classifier combined</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>with the Championship Algorithm.</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>Conditions for clustering components into words.</td>
<td>123</td>
</tr>
<tr>
<td>6.8</td>
<td>Example of iterative segmentation by adaptive threshold.</td>
<td>124</td>
</tr>
</tbody>
</table>
List of Tables

6.1 Results of the cross-validation testing procedure for autoassociators. 115
6.2 Computational cost for autoassociators and for the Championship
   Algorithm. ................................................................. 119
6.3 Computational cost of the MLP classifier and of the Champi-
   onship Algorithm. ......................................................... 121

A.1 Confusion Table for digits. Learning epoch = 1250. ............... 129
A.2 Confusion Table for uppercase letters. Learning epoch = 900. . . . 130
A.3 Confusion Table for lowercase letters. Learning epoch = 1250. . . 131
A.4 Experimental results for all classes, using autoassociators. ..... . 132

B.1 Confusion Table for digits: rej_Thres = 5.4, d = 0.4. .......... 133
B.2 Confusion Table uppercase letters: rej_Thres = 5.4, d = 1.2. .... 134
B.3 Confusion Table lowercase letters: rej_Thres = 4.6, d = 0.8. .... 135
B.4 Experimental results for all classes, using the Championship Al-
   gorithm. ........................................................................ 136

C.1 Experimental results for all classes, using MLP classifier combined
   with the Championship Algorithm. ................................. 138
Chapter 1

Introduction

One of the challenge of computer science is concerned with the development of tools capable of a more and more complete interaction with the real world. An aspect of such a problem is represented by the analysis and the comprehension of the information contained in signals of different nature, for example speech signals or images. It can be viewed as a general problem of pattern recognition: it requires the acquisition of the signal, its representation as a pattern and the use of a task-specific knowledge, codified in a symbolic or sub-symbolic fashion, in order to extract information. A particular attention is addressed to image understanding, and topically to images derived from document acquisition in electronic format.

Nowadays, in fact, paper is the main medium for exchanging information in literary, scientific or commercial field. The growing need of sharing information throughout work and research communities, joint with the development of new technologies for digital information diffusion, has increased the demand of tools for automatically converting information hold on paper into digital one.

Such a topic is particularly interesting in order to create databases of information extracted from documents or systems capable of storing and retrieving information from databases of intrinsically multi-media documents, where paper is just one of the media. We are still far from the “zero-paper option” and probably such a scenario is never to be achieved. Anyway, as far as it will be necessary, the problem of converting paper information into numerical one holds an important role.

In this optic the problem of the automatic document reading has raised a growing interest in the last few years, for example in the field of office automation. Many firms have to handle a large amount of documents every day with a great waste of time and work. One way to face the problem is to acquire the information by converting each document into an electronic format, but the information that they contain is difficultly retrieved in a selective fashion. Hence the need to create a database, extracting data from
documents; a database represents a more nimble consulting tool and ensures a better data rationalization and compression with respect to document storage in the electronic format. When the amount of data to acquire is considerably large, the automatic data entry in the database results particularly desirable. Therefore, a great effort is addressed to build up systems which are able to understand as wide domain of documents as possible.

The whole of the procedures intended to extract and understand information printed on documents goes by the name of document processing. From a theoretical point of view, document processing results intriguing since it deals with problems of image analysis, namely low-level image processing techniques, and, from a higher level of investigation, problems of machine learning, domain modelling, knowledge representation, semantic labelling and statistical reasoning.

The design of systems for particular reading tasks on single classes of documents with a fixed layout has reached a consolidated level of maturity [1], [2], [3], [4].

A more limited number of works deals with the problem of system flexibility, which is one of the principal goals of document processing. It is concerned with the task of multi-classes of document image understanding. The systems aimed at flexibility usually use a knowledge-based approach to document processing. The knowledge on domain is used directly in the understanding phase without a predefined goal of reading, in order to set up an open system which can be adapted to read different types of documents [5], [6], [7].

Other systems are intended to realize a compromise between reliability and flexibility. They aim at an enlargement of the domain of documents to be processed, preventing a great decrease of reliability. They are intended to individuate a number of classes in a document domain of interest so to limit the layout variability a system has to face. Moreover, they introduce the possibility of recognizing the class a document belongs to and to perform the document understanding by tools well-suited for the recognized class of documents [8], [9], [10], [11], [12], [13].

The architecture proposed in this work is placed in this middle policy of development. It is an attempt to design a document processing framework for a multi-class domain of documents, that is more flexible than those oriented to the comprehension of a single type of layout, limiting the loss of performances that the enlargement of the goal can determine. The system is based on a general knowledge, formalized by a semantic network which describes the document domain under consideration. Such a knowledge attempts to capture some structural and logical similarities among the classes by describing the logical relationships among document parts and their physical constraints.

This knowledge is used as a procedural scheme for processing documents of a known class, or for a semi-automatic construction of a specific model for each unknown class of documents of the domain. Then, the new model can be used during the phase of document understanding for further documents belonging to the new class.

Any document processing architecture assumes to trust to a reliable OCR (Optical
Character Recognizer) system, which performs the final task of reading a text from a data field, whose semantic is already assigned. The systems that can be found in literature usually rely upon commercial OCR systems. The disadvantage of such systems is that, generally, they are not well suited for particular classes of documents and for their specific fonts. This fact can sometimes determine a certain weakness in character recognition, making the reading task fail.

In order to ensure a high degree of reliability to an OCR system, to be adapted to a particular domain of documents, a font adaptive modular OCR system has been developed.

It is a neural-based architecture system that can be used for different domains. In fact, it is provided with learning capability with respect to new character examples. It can be obtained a new specific package for a specific font domain.

This work is organized as follows.

In § 2 the formulation of the automatic reading problem and an overview of the document processing systems that can be found in literature are presented.

In § 3 the overall structure of the architecture for a multi-class document processing system is illustrated.

In § 4 the framework for the proposed architecture for processing document of a multi-class domain, represented by a semantic network, is described. Particularly, the semantics and the pragmatics of the network are shown.

In § 5 an application of the proposed architecture to the invoice domain, namely a typical multi-class domain, is presented.

In § 6 the description of a font adaptive modular OCR system, joint with some experimental results on character recognition (§ A, § B), is presented.

Finally in § 7 some conclusions are reported.
CHAPTER 1. INTRODUCTION
Chapter 2

The document automatic reading problem

2.1 Generalities

The development of document automatic reading systems is a topical problem for office automation as well as in all those fields where data hold on paper need compression, translation or transformation to different media.

The automatic data acquisition from a paper document is a complex task that involves low-level procedures related to image processing, and higher level procedures related to the labelling of the extracted parts of a document.

Any document in fact is a collection of objects, basically rectangular areas, which may contain characters, blocks of text, figures, graphic elements, and which can be described at a geometric level, so to represent the geometric structure, called also layout structure of a document, or at a logical level, determining the logical structure.

According to [13], [14] and [15], the geometric structure of a document therefore is the collection of the extracted objects, obtained by the repeated division of the content of the document into increasingly smaller parts (basic objects), on the basis of the presentation. An object of the layout structure is also called physical object.

Such objects may be represented in a hierarchy (Fig. 2.1) with respect to the possibility of grouping together basic objects into a composite object, as well as grouping composite objects of lower level. Each object is described by its physical features (as coordinates, type of enclosed content, qualitative position with respect to other object, dimensions).

The logical structure of a document, on the other hand, is the collection of the extracted objects, obtained by the repeated division of the content of the document into increasingly smaller parts, on the basis of the human-perceptible meaning of the content.

Such objects (logical objects) also may be represented in a hierarchy (Fig. 2.2), and
each of them is described by its semantic attribution; the logical objects may also be described by the keywords contained in the text, and some other properties as qualitative relationships among objects or formatting properties.

From this point of view, the document reading task is concerned with two different levels of processing: the extraction of the layout structure, which is called document analysis and the assignment of a semantic label to each object of the layout structure, obtaining the instantiation of the logical objects of the document: such phase is called document understanding.

Basically document analysis represents all the low-level processing procedures which are intended at the extraction of basic or composite objects, associated to their physical feature description. Such functions are concerned with the extraction of primitive graphic elements, basically connected components ([16]), and, on their basis, segments and not segment components, usually characters, words, rows of text and other graphic elements.

On the other hand document understanding represents the set of algorithms which are
intended to assign a semantic label to the physical objects extracted during the document analysis phase. Basically such procedures provide the information an object contains with a semantics, thus identifying the logical structure of a document.

A possible approach to document understanding is that of using an optical character recognizer (OCR) in order to read the content of each physical object and then deriving its semantic.

However such an approach usually has a high computational cost. Therefore the most widely approach to document understanding is to make use of the possible relationships which can be established between layout and logical structure.

As well as relationships between layout objects and between logical ones can be described (for example the position between two different physical or logical ones), similarly relationships between the physical structure and the logical one can be established. Such relationships allow a system to recognize a logical object among the physical objects extracted during the document analysis phase, only using layout features.

Such relationships are usually described and contained in an a priori knowledge which is called the model of a particular type of document.

A model can be expressed in terms of a deterministic knowledge, namely a set of
rules or constraints on the objects which allow to locate them univocally, or in terms of a probabilistic knowledge, which describe the possible positions or physical constraints of an object with a degree of uncertainty.

The matching between the instance of a document and its model provides each extracted physical object with a semantic label. Thus the logical structure of document and the model result instantiated.

The last phase for the entire document recognition and acquisition of data is represented by reading the content of the objects by an OCR system.

Nevertheless, when the set of documents that is to be processed is wide, and it is characterized by a sensible variety of layout structures or logical structures, the understanding phase may become a difficult task. In fact, if a domain is characterized by some variability in the layout structure or in the logical structure, a single model may not be sufficient to describe the entire domain. Therefore it may not guarantee a high reliability in the understanding phase.

Thus an intermediate phase may become necessary: such a phase is called document classification. It represents the attempt to simulate the human behaviour of grouping documents into classes from a perceptive point of view, according to the subject, the layout or the logical structure, or by reading the content of particular parts of the document.

This procedure allows a system to associate a different model to each different class. Such possibility can greatly improve the reliability of the understanding phase. In fact, grouping documents of a domain into classes allows to reduce the variability of layout or logical structure that has to be described by a model.

In literature we can find both systems which are characterized by only the document analysis phase followed by the document understanding one [4], [5], [1], both those which perform the three phases of document analysis, classification and understanding in this order [12], [13], [8], [9], [10]. The choice of using a system architecture or the other one depends on the characteristics and the variability of the domain of interest, and on the requirements of the project in terms of flexibility and reliability. The two schemes of document processing are reported in Fig. 2.3.

In § 2.2 a possible sub-division of documents into domains and classes is described, while in the § 2.3 a number of architectures for document reading systems are discussed. Such architectures will be evaluated in § 3.1 on the basis of the characteristics of the proposed class division they are applied to.

2.2 Domains and classes of documents

The universe of documents is characterized by a wide variety of typologies, which correspond to different layouts and logical structures.

A domain can be defined as a group of documents which can be clustered with respect
2.2. DOMAINS AND CLASSES OF DOCUMENTS

Figure 2.3: The two schemes of document processing.

to the subject: for example journals, papers, tax forms, business letters, invoices, check forms and so on, are different domains.

Each domain can also be characterized by some features which can make effective the automatic domain classification. Such features can be concerned with the physical, or the logical structure. For example a domain can be associated to the presence of a particular object (for example the abstract for a paper), the presence of a particular text or some other features (for example the dimension for a check form), so to obtain a one-to-one mapping which is sufficient to identify the domain of a document instance.

Formally, if we introduce in the universe of documents an equivalence relation $R_d$ that considers two documents in relation if they share the same subject (for example the payment of taxes for a tax form, the payment of commercial transactions for an invoice, and so on), we obtain a partition of the universe of documents in equivalence classes, which we call domains.

If $\mathcal{U}$ is the whole universe of documents, let $D_i$ be a document ($D_i \in \mathcal{U}$), then

$$[D]_{jR_d} = \{D_u, D_v \in \mathcal{U} : D_u R_d D_v\}, \quad j = 0, ..., T - 1;$$  \hspace{1cm} (2.1)

is the $j^{th}$ equivalence class related to $R_d$, in the universe of documents, and $T$ is the number of the classes.

Furthermore, we can indicate as $Q_{R_d}$

$$Q_{R_d} = \mathcal{U}/R_d$$  \hspace{1cm} (2.2)
the quotient set for the relation $\mathcal{R}_d$. It represents the set of equivalence classes whose elements are the single domains: for example the domain of journals, of tax forms, of invoices and so on (Fig. 2.4).

![Diagram of the universe of documents divided into domains.](image)

**Figure 2.4:** The universe of documents divided into domains.

Documents belonging to a particular domain can be further characterized by different layout or logical structures. Anyway, especially in commercial form field or in the field of business letters, it is possible to cluster documents belonging to a specific domain into classes, which are characterized by physical or contextual similarities.

A *class* can be defined as a group of documents whose elements are characterized by the position invariability of each object (the physical object and its semantic label) in terms of qualitative relations with respect to other objects or in terms of coordinates.

The automatic class identification appears easier with respect to the corresponding task for document domains. In fact a class is strictly characterized by the position and the semantic invariability of the objects. Therefore, if no other specific features can be associated to a particular class, its identification can be based on the layout structure itself, and, if it is not sufficient, on the identification of some semantics.

Formally, in a document domain an equivalence relation $\mathcal{R}_p$, which considers two documents in relation if they present the same physical and logical structure in terms of object semantics and object coordinates, can be introduced.

Similarly, an equivalence relation $\mathcal{R}_l$ can be introduced. It considers two documents in relation if they present a different physical structure but the objects which have the same
2.2. DOMAINS AND CLASSES OF DOCUMENTS

semantics present the same mutual qualitative position ("left-of", "right-of", "above", and so on).

With respect to the equivalence relation just introduced, a partition of a document domain in equivalence classes can be obtained:

1. **Physical classes**
   which contain documents grouped by the \( R_p \) relation;

2. **Logical classes**
   which contain documents grouped by the \( R_l \) relation.

![Figure 2.5: Two examples of documents belonging to different physical classes but to the same logical one.](image)

Basically, the logical objects of two documents belonging to the same logical class can occupy different positions in terms of coordinates, but their mutual positions have to be the same (see Fig. 2.5).

On the contrary, the logical objects of two documents belonging to the same physical class must occupy the same position in terms of coordinates.

If \( D_h \) is a domain of documents, let \( D_i \) be a document \((D_i \in D_h)\), then

\[
[D]_{R_p}^j = \{D_i, D_m \in D_h : D_i R_p D_m\}, \quad j = 0, ..., N - 1;
\]  

(2.3)

is the \( j^{th} \) equivalence class of the \( N \), related to \( R_p \), in a domain of documents. Similarly

\[
[D]_{R_l}^k = \{D_r, D_s \in D_h : D_r R_l D_s\}, \quad k = 0, ..., M - 1;
\]  

(2.4)

is the \( k^{th} \) equivalence class of the \( M \), related to \( R_l \), in a domain of documents. Therefore a logical class can contain different physical classes (2.5)
\[
\{[D]^{k(0)}_{R_p}, [D]^{k(1)}_{R_p}, \ldots [D]^{k(i)}_{R_p} \ldots \} \subseteq [D]^{k}_{R_l};
\]

(2.5)

Figure 2.6: The relation between a logical class and its physical sub-classes.

Furthermore, we can indicate as \( Q_{R_p} \) and \( Q_{R_l} \),

\[
Q_{R_p} = D/R_p
\]

(2.6)

\[
Q_{R_l} = D/R_l
\]

(2.7)

the quotient sets for the relations \( R_p \) and \( R_l \), respectively.

Hereinafter all the references to domains or to logical or physical classes are related to the equivalence relations \( R_d, R_p \) and \( R_l \).

### 2.3 Types of architectures for document processing

Literature about document understanding has grown impressively in the last few years. It seems not so easy to find one’s bearing among the great deal of works related to different domains which use different approaches towards the reading task. However, a classification can be attempted with respect to the types of architectures used, which can be classified into three different architecture classes: *Closed architectures, Open architectures* and *Semi-open architectures*.

In each kind of architecture the two different phases of *document analysis* and *document understanding* can be identified.

*Closed architectures* ([4], [1], [2]) are types of system which are oriented to the automatic extraction of information from a single class of documents in the sense of § 2.2, or to use different models for different classes a domain consists of.
2.3. TYPES OF ARCHITECTURES FOR DOCUMENT PROCESSING

Generally in a Closed architecture the phase of document analysis extracts the physical objects a document consists of, then the phase of document understanding uses a model for each class of documents. In a domain of interest, a model, which contains the mutual positions of the objects and their semantics, can be sufficient to assign a semantic label to each extracted physical objects of each physical class, if more physical classes are subsets of a single logical class. Otherwise for each physical class, a model, which contains the coordinates of each object and its semantic, is necessary during the understanding phase to univocally locate an object and to assign it a semantic label.

Open architectures ([5], [6], [17]), on the contrary, are specifically designed to process different domains of documents or, more strictly, different classes of documents belonging to the same domain. They are characterized by no prior knowledge about any specific domains or classes of domains, but contains a set of image processing tools for document analysis and a set of problem-solving strategies that are applicable to documents. A general domain independent knowledge or a knowledge related to each single domain is considered part of the input and is applied directly during the understanding phase, without any other consideration about the class the current document belongs to and thus without a predefined goal of reading. The knowledge about a domain or of a class is considered as an input to the system, when the domain or the class to be processed is a priori identified.

Semi-open architectures ([8], [9], [10], [12], [13], [11]) express a middle policy of development among the foregoing ones: after the analysis of the document, such systems aim at an explicit document instance classification and during the understanding phase

![Figure 2.7: A domain of documents divided into classes.](image)
use a specific model for each class, in the sense described in § 2.2, with a deterministic location of the objects, or for classes which have not the fixed structure as defined in § 2.2, with a probabilistic semantic attribution to the extracted objects. In case an instance of document doesn’t belong to any of the known class, a new model has to be construct, usually by user interaction. Such a model contains object qualitative positions or object coordinates, and their semantics.

In Closed architectures, the problem of document classification is implicit, since usually is dwelled upon the user who feeds the system with documents belonging to the same class. Eventually, it reduces to a procedure which verifies weather an instance belongs to the class of interest or not. In the letter case the instance is rejected. In such kind of architectures only the phases of document analysis and of document understanding are present.

Similarly, Open architectures present only the document analysis and the document understanding phases. The phase of document classification is a priori performed, providing the system with documents belonging to a particular domain, or it is reduced to verify whether an instance of document belongs to the domain of interest, otherwise it is rejected [5]. Such systems contain a general knowledge, described by probabilistic physical constraints or logical relationships between objects of a particular domain. Therefore, if the domain is structured in classes, in the sense described in § 2.2, the understanding phase can be performed by a probabilistic semantic attribution of the extracted physical objects.

On the other hand in Semi-open architectures, as previously discussed, the phase of document classification is crucial ([12], [13], [8], [9], [10]), in order to associate the related model to the instance of known class or to construct a new model for a unknown class. In

---

**Figure 2.8:** Closed and Open-architecture schemes.

(Lam, 1994)
fact, when the membership class of a document is known before the phase of understanding, the variability of the logical and the physical structure is significantly reduced. In this case, the possible layout and logical structures, which a document of a class may assume, are limited with respect to the logical-layout relationships allowable for a document of the whole domain.

Therefore the problem of document understanding can be greatly simplified by classifying instances of documents.

Graphically the three kinds of architecture can be described as in Fig. 2.8 and in Fig. 2.9.

![Diagram](image-url)

Figure 2.9: A Semi-open architecture scheme.

Examples of the three kinds of architectures are discussed in § 2.3.1, § 2.3.2 and § 2.3.3.

### 2.3.1 Closed architectures

In [4] a system for the analysis and the understanding of the GIRO check forms used in Switzerland is presented. The goal is to recognize the name and the address of the receiver, his financial institution and account number, and the amount of money. Such objects, namely fields which contain information, may be printed at variable locations, in different formats and in various fonts, therefore such a form have a flexible layout structure in contrast with those forms in which information fields are printed at predefined positions. However, this kind of form is characterized by the qualitative position invariability among the objects, since they are printed according to a set of rules. Thus, the GIRO check results a logical class of document and the understanding phase can sufficiently use a model which contains such rules in terms of qualitative positions among objects.
In [4] the problem of document processing is divided into three related subproblems: 1) layout analysis, 2) OCR, 3) post-processing with error correction.

Layout analysis locates text and graphics, therefore it involves both the phase of document analysis and of document understanding, the OCR system identifies the contents of text regions, post-processing verifies the OCR outputs with respect to a dictionary. The relationships between the physical structure and the logical one is represented by a model formalized by a graph; it is composed by nodes and edges, which link the nodes together. In [4], nodes are associated with the objects of interest in the document and edges represent spatial relationships between objects. The objects can be classified into three categories (Fig. 2.10):

- **GRAPHICS**: a graphic object can be a rectangular box or a straight line;
- **TITLE**: title objects are predefined string of characters and they function as keywords;
- **INFORMATION**: strings of characters entered by the sender of the check.

Moreover there are three kinds of spatial binary relationships among the objects: *Above, Left-of* and *Inside*, with the intuitive meaning. Unary relationships, as well as attributes, describe intrinsic properties of an object: for example the length of a line, the content of a TITLE object. No absolute position of any object is fixed on a check. The spatial relation between different objects on a check must follow the model graph (Fig. 2.11).
The recognition process is performed through the primitive extraction (lines and keywords) and, with their respect, through the location of the TITLE objects; the matching with the model, that establishes a correspondence between a model (Fig. 2.11) and a data graph, gives the positions of the related INFORMATION fields and gives them the semantics. The TITLE object knowledge can be divided into two parts: the structural and the statistical one. The structural knowledge is related to the size of character, the total length of the string and the forms of rectangular bounding box. The statistical one is related to the information contained in the string of characters. The statistical knowledge can be represented by the histogram values of the vertical projection of black pixels (also called vertical profile).

The identification of a TITLE object is obtained by two tests about the structural knowledge and the statistical knowledge. They are based on distances between the related TITLE vertical profile and some reference profiles. The tests are considered correct if the distances are smaller than a predefined thresholds.

The low-level processing is concerned with binarization and segmentation.
Image binarization is performed by the automatic threshold selection algorithm proposed by Otsu [18]. This algorithm first computes the histogram of the gray-level image and then determines the binarization threshold. The threshold determination is based on the maximization of the inter-class variance assuming that the image is composed by only two classes, foreground and background pixels.

The segmentation of the image is based on the $X$-$Y$ tree decomposition procedure, which exploits the fact that most document images have a vertical and horizontal structure. Recursive cuts are performed, evaluating horizontal and vertical projections of black pixels. The result of the $X$-$Y$ tree decomposition is a tree whose nodes represent rectangular zones, and the leaves the zones which are not further decomposable (Fig. 2.12).

![Document Image Tree Representation](image)

Figure 2.12: A document and its tree representation.

The limit of such a method is represented by skew sensitivity, but balanced by a higher computational speed with respect to connected component procedure.

Another Closed architecture for the analysis and understanding of financial document is proposed in [3].

In this work two different subsystem are described: a fixed document processing subsystem based on staff line approach and a complex document processing subsystem operating in a form description language (FDL).

The first subsystem (fixed document processing subsystem) is aimed at extracting the filled data from Canadian bank cheques. Since both their geometric and logical structures are fixed, it is possible to use a fixed (closed) system to process this kind of financial documents. Such a subsystem extracts the staff lines and assigns to the information over them the semantic label with respect to a model. In such a model the label of each staff line, ordered with respect to their position, is provided.

The second subsystem (flexible document processing subsystem) is intended to process
 financial documents with different layouts or possess complex structures. It is still a Closed architecture system since there exists a model for each class of document characterized by the position invariability among the objects (here called items) in terms of coordinates, even if the model describes the positions of the items by qualitative relationships. Prior to form understanding, the form description written by the FDL (Form Description Language) is read into the system and interpreted to become a form structure. In this system the phase of document classification is implicit and dwelled upon the user who describes and selects the model for the current form.

Then the purpose of the mapping stage is to find the actual locations and relations of lines and bands in the form and establish a correspondence between the form description and the form structure.

For each physical and logical layout a model description formalized by the FDL is proposed.

The FDL presented in [3] includes two parts:
- FSD (Form Structure Description) which describes the structure of the document;
- IDP (Item Description) which describes the items in a document.

Let \( \Omega_F \) be the form space

\[
\Omega_F = (\sum \times \sigma_D \times \sigma_G \times \sigma_P \times \Delta)
\]  

(2.8)

where:
\( \sum \) is a finite set of graphical objects which may be lines \( \{L_1, ..., L_l\} \) and bands \( \{B_1, ..., B_m\} \);
\( \sigma_D \) is the set of the directional property of \( \sum \), \( D_V \) represents the vertical direction, \( D_H \) the horizontal one and \( D_S \) the slant direction;
\( \sigma_G \) describes the geometrical characteristic of \( \sum \) (\( G_{TC} = \text{Thick}, G_{TN} = \text{Thin}, G_{TM} = \text{Between Thin and Thick} \) of \( \sum \));
\( \sigma_P \) is a vector of the position property of \( \sum \) (\( P_V = \text{Vertical position of the left top most point of a graphic object}, P_H = \text{Horizontal position of the left top most point of a graphic object}, P_L = \text{Length of a line or a band} \));
\( \Delta \) is a number for the identification of a specific graphic object.

Therefore:

\[
FSD \doteq (\sum \oplus \sum \oplus ... \sum), (D_V \oplus D_H \oplus D_S), (G_{TC} \oplus G_{TN} \oplus G_{TM}), P_V, P_H, P_L, \Delta)
\]

IDP includes two parts:

- \( \text{ILD} = \text{Item Location Description} \);
- \( \text{ICS} = \text{Item Component Selection} \).
Let \( \alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \) be a set of items and \( \sum = \{\sum_1, \sum_2, \ldots, \sum_n\} \) be a finite set of graphic objects. Suppose there exist a set of relations between items and graphic objects:

\[
\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}
\]

\( M_{ID} \) is called an Item Description Matrix:

\[
M_{ID} = \begin{cases} 
\Gamma_l & \text{if } (\alpha_i, \sum_j) \in \Gamma \\
0 & \text{if } (\alpha_i, \sum_j) \notin \Gamma
\end{cases}
\]

which satisfies the following condition:

\[
\forall l \ (\Gamma_l = (\alpha \bigcup \sum_j)), \quad \mathcal{R} = \{R, L, A, B\}
\]

where \( R, L, A, B \) represent Right, Left, Above and Below. The matrix \( M_{ID} \) therefore represents the relations between items \( \alpha_i \) and graphic objects like lines \( L_i \) or bands \( B_i \), and it is used for the semantic attribution of each item, individuated by staff lines.

Finally, several components contained in an item, which are usually words or number strings, may be selected by an item component selector (ICS).

Each item \( \alpha_i \) may be represented by an \( m \times n \) matrix:

\[
\alpha = \begin{bmatrix}
w_{11}^i & \cdots & w_{1m}^i \\
w_{21}^i & \cdots & w_{2m}^i \\
\vdots & \ddots & \vdots \\
w_{nm}^i & \cdots & w_{nm}^i
\end{bmatrix}
\]

where \( n \) is the number of lines the item \( \alpha_i \) occupies; \( m \) is the number of words in a line of the item \( \alpha_i \).

The component selector \( ICS(jk; rs)\alpha_i \) selects the elements \( w_{jk}^i, w_{rs}^i \) of \( \alpha_i \).

In such a system a kind of learning is provided. The parameters of lines extracted during the document analysis are fed-back to the mapping stage: a set of new parameters are generated and employed in the next cycle of the mapping. As a result, the system processes an adaptability and will become more accurate as more samples are processed.

If \( n \) is the number of document graphic object and each graphic object consists of \( m \) parameters, let \( x_k^i \) be the current value of the \( k^{th} \) parameter of the \( i^{th} \) graph (lines or bands), and \( \sigma_k^i \) be the value of the corresponding parameter described by the FSD. The error between them is computed as:

\[
\epsilon = \sum_{i=1}^{n} \sum_{k=1}^{m} [x_k^i - \sigma_k^i]^2
\]

The learning rule used to modify the value of each parameter is given by:

\[
\sigma_k^i(t+1) = \sigma_k^i(t) + \alpha \times [x_k^i - \sigma_k^i(t)]
\]

(2.13)
where $\alpha$ is the learning rate.

In the understanding phase the model related to the class of the current document is acquired. With respect to the model of the selected class, the structural knowledge of the current instance is extracted. If $\epsilon \geq \tau$, where $\tau$ is a threshold, the parameters of the model are updated according to the 2.13.

In [1], [2] a Closed architecture for processing forms of a known class is presented. It uses a model based on Attributed Relational Graphs [19] that gives an accurate and flexible description of the form class. Form registration and location of information fields are performed by using algorithms based on the hypothesize-and-verify paradigm [20].

The approach is related to the one described in [21], as the information fields can be identified only after having recognized the corresponding instruction field.

The model, based on attributed relational graphs, performs form registration using algorithms based on the hypothesize-and-verify paradigm. Such a model is based on the assumption of dealing with forms of a given class that, however, has a flexible structure. The model of such forms is obtained by defining some registration landmarks, that can be both lines and keywords (considered in the model as instruction fields) implemented by nodes with attribute. The mutual position between landmarks is implemented by arcs with the attribute. Moreover, in the reference model of the form an error margin that specifies the maximum allowable displacement for an image of an incoming form is defined. This value is taken into account to deal with the skew and the translation of the form.

Such system deals with a physical class of documents (§ 2.2), represented by a graph which contains the mutual positions of the objects in terms of coordinates. However, it introduces a certain flexibility, in the sense of allowing the user to modify the model when the system has to process different physical classes of documents. Therefore, only the phases of document analysis and understanding are present, since the class of the document is a priori known.

### 2.3.2 Open architectures

Systems oriented to read multi-classes of document images show a considerable flexibility, but a certain weakness in the understanding phase [5], [17], [6], [7]. These systems propose a knowledge-based approach to document processing, which is used directly in the reading task for different classes of documents. Such an approach provides these systems with a considerable degree of flexibility, however it produces a probabilistic semantic labelling of physical objects, with loss of reliability.

In [5] an open system architecture for document image understanding is proposed. The
system is intended to process different domains of documents by using a set of problem-solving strategies and image processing tools that are applicable to documents. A prior knowledge about any specific domain is present too, but it is considered as a part of the input.

Such a knowledge contains world features (Document Independent Knowledge (DIK)) for example the fact that the order of identifying two objects gives preference to the one on top or to the left, and specific ones for each domain of documents (Document Specific Knowledge (DSK)). These two kinds of knowledge are defined and developed independently. The DSK is formalized by a declarative knowledge for each domain, based on a hierarchy of objects which represent the regions of interest (ROI) of the domain. The hierarchy corresponds to a tree where the objects are the nodes and the links between nodes represent the taxonomy relationships among objects. Each object is described by a hierarchy of frames ([22]) which contain the features of each ROI in terms of spatial constraints, contextual relationships among regions and other information as cardinality, priority and obligatory of each region in the domain.

Given such a hierarchy of descriptive objects, a strategy planner derives a logical order which can be used as a guideline to locate objects from the image.

As far as the reading phase is concerned, it is assumed that the DSK and special tools have already been defined prior to reading. Thus, no classification phase is provided in this system. Firstly the phase of document analysis is performed by extracting primitives and globally the layout structure of the current document. On its base ROIs are located. A strategy planner, that derives an order among the objects, selects an object in the hierarchy to be the focus of attention. If an object is a non-terminal node in the hierarchy, its children at the lower level of the hierarchy will be pursued first; therefore the controller uses a depth first traversal scheme. With respect to DIK and DSK a number of candidate objects are selected according to the reading plan. Candidates for the objects are kept in the data space and the candidates are organized in the same hierarchical structures as the objects in the document model. The selection process of object candidates depends on the ranking of the candidates. Ranking is based on a credibility measure which reflects how well a candidate satisfies the constraints required for a particular object. To the candidate with the highest credibility the semantic label of the current object is assigned.

A quite analogous approach is described in [17] and in [6], where an architecture, applied to the domain of business letters, has been proposed.

In [17] firstly document is partitioned in text and non-text blocks. Then the semantic label is them assigned with respect to logical and physical constraints among the extracted blocks. Two kinds of knowledge are used: a form oriented knowledge, which corresponds to
2.3. TYPES OF ARCHITECTURES FOR DOCUMENT PROCESSING

the layout knowledge, and a function oriented knowledge, which corresponds to the logical one. Knowledge about document structure depends also on other resource as OCR and additional image processing. The two kinds of knowledge represent the Document Structure Knowledge Base, while the knowledge extracted from a current instance is called Document Instance Knowledge Base. The Document Structure Knowledge Base is a hierarchy of object described by a taxonomy of frames which characterize the function of each component and the hierarchy dependent relationships (is-a and has-a hierarchy, Fig. 2.13).

![Figure 2.13: An example of a region taxonomy which is used for functional components analysis.](image)

The result of interpreting a particular document is represented as an instantiation of the Document Structure Knowledge Base in the Document Instance Knowledge Base. In this system the phase of document classification is present. It is based on the recognition, by the OCR system, of specific objects which belongs univocally to a domain. After having classify the document, the system use the Document Structure Knowledge Base combined with the Document Instance Knowledge Base and the OCR to label each physical object without a specific model related to a particular domain or class of documents.

The system presented in [6] is designed for single-page business letters belonging to different logical and physical classes.

It is not assumed a one-to-one correspondence between physical objects and logical ones (here called functional components). Then labels of the functional components are attempted to be mapped to physical objects via a process whose rules include intra and inter-objects knowledge of various kinds.
According to Fisher [23] document units are characterized by three types of information: location cues, format cues and textual cues. A location cue is an inter-object information, while format and textual cues as for example keywords, text grammar identification, the frequency of a string of characters or a syntactical structure are an intra-object information. Rules are expressed with a simple description and in terms of sufficiency fashion, rather than as rigorous as possible. A threshold-based monotonic labelling procedure is used. If a physical object meets sufficient criteria to be assigned with a label, the label is assigned and the object is no more visited.

For the labelling procedure some “landmark” components are used. By exploiting the positional relationships of objects to these landmarks, we are able to greatly reduce the search space for functional components associated with these physical objects. Physical objects not yet labelled can be grouped in a single functional objects through a recursive process, till a new cycle produces no new labelling. In addition to physical and relational rules, the labelling function is supported by the recognition of object keywords which can be contained in a dictionary. The raw score for word searching is the number of characters in corresponding positions which match (up to the length of the dictionary entry), normalized by maximum of the candidate length and the dictionary length.

In [24] a document automatic reading system, which aims at the understanding of text portions of structured documents of a defined domain, is presented. Such a system is highly supported by knowledge since it contains a document model covering composition rules, geometric and lexical constraints, and a tool-box of document image analysis algorithms. Its architecture connects declarative knowledge about a document and the image analysis and so that the algorithms can be applied expectation driven. The tool-box describes each algorithm by its essential features, whereas the document model describes the geometric and lexical properties of an object as well as relationships between objects. Each object, either a document object or an algorithm, is represented by the semantic network language Fresco (Frame Representation of Structured Documents). The procedural aspects are represented by an inference and control algorithm: they are responsible for generating results and finding the best possible solution.

The basic structures in Fresco are: concepts, attributes, part, descriptions and constraints. Instances to a concept are generated during the document analysis. These objects consist of several substructures described in [25], [26], [27]. The subclass relation and the has-part relation are the only relations supported by Fresco. The knowledge about document objects is modelled by concepts, divided into two disjunctive subsets: layout concepts, which are represented in the layout model and logical concepts of a specific domain, which are represented in the logical model. Each concept can be described by an arbitrary set of attributes, parts and constraints representing geometric and lexical properties, layout rules and grammars. The basis of the document representation is the
2.3. TYPES OF ARCHITECTURES FOR DOCUMENT PROCESSING

layout model. It defines the general layout concepts like text-block, line or character, and general layout rules, like lines-left-aligned, words-on-same-baseline. On the other hand, logical concepts describe the knowledge about the specific domain of application. Since the layout knowledge is valid for any kind of documents, the layout model and the attached layout rules are predefined in Fresco. The layout concepts are described in different levels of abstraction: from the level of connected components up to the abstract level of text block. As well, logical concepts are represented by objects from the level of documents down to words and even characters. Both sets of concepts are linked together by subclass and part links. Since logical concepts are linked to layout concepts by a subclass link, all layout knowledge is inherited to a logical concept.

The goal of Fresco’s inference algorithm is to generate instances to layout and logic concepts, until specific text portions of a document are interpreted. The inference algorithm is divided into three phases which process the knowledge about structured document top-down and bottom-up: they are related to the selection of the logic concept, the selection of the possible instances by the selection of the algorithms with respect to the logical parts to be instantiated, and the definitive instantiation of the logical concept.

The first phase generates concept hypotheses top-down that have to be verified in the subsequent instantiation phase. The expansion process steps down recursively along the part link until all the layout concepts, related to the logical one selected, are entered.

The task of the second phase is to build the layout structure of the document: it performs the document image analysis generating instances to layout concepts in order to satisfy the hypotheses list. In each step more than one algorithm may be applicable. In such a case the algorithm with the fewest cost is selected. These algorithms are applied expectation driven, since the concepts expanded in the first phase provide knowledge about the properties of document objects to be analyzed.

In the third phase the instances generated by these algorithms of the tool-box are interpreted in terms of the list of expanded hypotheses. In order to evaluate the possible instances a set of statistically adapted classifiers or heuristic fuzzy set are used. An instance is evaluated combining the certainty values of its attributes and constraints. Since classifiers and fuzzy functions are used for evaluation of instances, the analysis is fault tolerant to a certain degree, dependent on the sharpness of decision, e.g. the fuzzy membership functions.

Also in [7] a knowledge-based approach to document processing is proposed. Such an approach is intended to encode knowledge about a new domain into the system as an input. The aim is to easier adapt a document reading system to changing requirements or to new application domains. Usually such a knowledge describes document parts, their logical relations, their constraints and the textual information that can be used to identify the parts. This knowledge can be used for different analysis task for a class of documents:
labelling all layout objects or to identify only some of them. System input may be an ASCII file or a paper document. In a preprocessing step an internal representation of the given text including its layout structure is created. The main components of the system are: the analysis component, task and strategy definition, layout structure, knowledge base and hypotheses. The main task of the analysis component is to build and to settle hypotheses about the meaning of the elements of the input document. The task definition contains information about the class of the document in input, whether it is known or not and which parts of the documents are to be searched for. The layout structure contains the result of preprocessing, that is a special representation of the input document consisting in words, lines of words and blocks of words.

The hypotheses component contains a description of the blocks with the probability values related to a part to be search for. The language used for the knowledge description should be declarative and easy modifiable. In [7] a frame-based language is chosen because of the built-in inheritance concepts and the possibility for integration of other kinds of concepts like predicate logic. In this case the standard frame concept is extended by descriptions of parts (part-of hierarchies), uncertainty and relations. For the analysis of different problems, task and strategies have to be changed and the knowledge base has to be adapted to the application domain.

2.3.3 Semi-open architectures

In this paragraph a number of Semi-open architecture are reported.

In [12] a system for multi-class domain of documents processing is presented. It is based on a knowledge formalized by Horn clauses. It allows to label each layout object with its semantic label (document understanding), starting from document physical knowledge (document analysis) and from the class the document belongs to. The problem that is presented is divided into two different steps: the acquisition of classification or labelling rules and their use for document classification and understanding.

The approach of learning rules is based on machine learning techniques in order to directly acquire classification or labelling rules from a set of training documents labelled by the user.

Since the aim of the system is to handle several kinds of documents, document understanding becomes a difficult process due to the different logical-layout relationships met in each kind of documents, as discussed in § 2.1. Thus the intermediate step of document classification is necessary. It represents the identification of the particular class the document belongs to. Layout information could help to recognize the class of a document
2.3. TYPES OF ARCHITECTURES FOR DOCUMENT PROCESSING

when there exists a definite set of relevant and invariant layout characteristics.

In [12], [13], given a set of documents whose page layouts have already been analyzed and assumed that the user-trainer has already labelled some layout components according to their meaning, the problem is that of learning some rules that allow the correct labelling of layout components to be performed.

The problem of learning rules for document understanding or the problem of understanding itself can be strongly simplified when the class of documents has already been identified. The problem of learning rules for document understanding is still more complex than the problem of learning recognition rules for document classification. In fact, in document understanding, concepts to learn refer to a part of a document rather than to the whole document, and since parts of documents may be related to each other according to logical-logical relationship, this leads to the problem of mutually dependent concepts (or contextual rules).

The problem of document understanding is a particular case of labelling problems, in which the correct label can often be assigned to a part of a complex object only by taking into account spatial relationships with other parts whose labels are already known. Such an assumption allows to generate more accurate and simpler rules. The structure of concept dependencies can be represented by means of a directed acyclic graph. A knowledge representation language, based on Horn clauses, in order to describe a page layout and the learning system FOCL is be used.

In [12] each instance of an object is represented as a ground Horn clause in which different constants represent different layout objects of one or more documents. In the machine learning literature it is made an implicit assumption that concepts are independent. In [12] such an assumption is not supposed to be the correct way to solve the problem. The dependence assumption among objects is supposed to make easier the recognition task, is made the learning problem is concerned with concepts as well as dependencies between concepts. As a natural consequence of concept dependency, instances of dependent concepts are dependent themselves. When concept dependencies are intrinsically acyclic, as in many problems of object labelling, a dependence hierarchy can be used (Fig. 2.14).

In [12] the learning problem is reduced to the problem of object labelling.

In general the complex object \( O \) can be decomposed in a set \( U = \{u_1, u_2, ..., u_n\} \) of units each of which can be named with a label \( l_i \) taken from a set \( L = \{l_1, l_2, ..., l_n\} \) of labels. Each unit \( u_1 \) is described by a set of attributes \( A = \{a_1, a_2, ..., a_p\} \) and by a set of relations \( R = \{r_1, r_2, ..., r_q\} \). The aim is that of assigning the right label to some (or all) units of \( O \). In [12] structured objects are represented in first-order logic. For instance, assuming that all relations in \( R \) are binary, \( O \) can be represented as a conjunction of (typically positive) literals involving binary predicates:

\[
a_1(u_1, c_{11}) \land ... \land a_1(u_n, c_{1n}) \land ... \land a_p(u_n, c_{pn}) \land r_1(u_1, u_2) \land ... \land r_n(u_{n-1}, u_n)
\] (2.14)
where $c_{ij}$ are constants representing values of the attributes, while $u_i$ are constants used to denote each object.

Then given a set of instances of a structured object for which some or all units have been labelled, the learning problem is that of learning rules that allow for labelling their units. Labelling starts with minimally dependent concepts. When the minimally dependent concepts have been learned, it is possible to learn those concepts that depend directly on them.

In [12] such a learning scheme is applied to document understanding. In this case learning begins from minimally dependent concepts too. The case-study is represented by business letters and the logotype is chosen as minimally dependent concept. FOCL is used to generate the rule for the concept logotype using extensionally or intensionally predicates.

Similarly, according to the hierarchy, the rules for the concept sender and the concept reference No. are generated. The rules for the concept date are generated by the rules of logotype, sender and reference No..

After having acquired rules the phase of document classification and understanding are performed.
2.3. TYPES OF ARCHITECTURES FOR DOCUMENT PROCESSING

The result is a flexible system that leads to a probabilistic semantic labelling of the objects. However in [12] the presence of document classification is intended to reduce the variability of documents to be described by a model, improving the reliability of object labelling.

In [9] a knowledge-based system for the identification and the labelling of regions in documents, in particular business letters, belonging to different physical class, is proposed. The system is based on a modular knowledge representation structure, called Geometric Tree, whose intermediate nodes represent classes of physical structures, in the sense of § 2.2; each class is refined in the lower levels till the leaves of the tree. Leaves of Geometric Tree represent the model of a concrete specific class of layouts for the domain of documents under consideration, and contains the physical coordinates of the objects and their semantic labels.

In order to describe the spatial structure of a letter, the page is divided into smaller rectangles by vertical and horizontal cuts. Cuts are placed so that they do not intersect with text or graphics. Cuts are represented with a notation that indicates their type (H (horizontal), V (vertical)) and the coordinate (vertical and horizontal, respectively) in terms of percentage of the entire page.

In addition to the structure model, a statistical data base (SDB) is used. There are stored the examination results of a few hundred business letters. During the understanding phase the system attempts to match the physical layout of the current document with structures contained in the Geometric Tree, executing a depth traversal. Classifying and understanding a given document amounts to finding a path in the Geometric Tree from the root to one of its leaves.

Starting at the root, in each step the current document is matched with the two layout sub-classes of the current node. The degree of similarity between the layout of the given document and the nodes in the model have to be quantified through a confidence value $0 \leq v(x) \leq 1$ which represents the cut validation function with respect to the position of the current cut in the model. If $l_1$ and $l_2$ are the boundaries of the area to be partitioned, and $c$ is the position of the current cut in the model, we have:

$$v(x) = \begin{cases} 
  f_1(x) \ast r_1(x) & l_1 \leq x \leq c \\
  f_2(x) \ast r_2(x) & c \leq x \leq l_2
\end{cases}$$

where:

$$f_i(x) = 1 - \frac{(x-c)^2}{(l_i-c)^2}$$

defines the entire validation function for cut shifting within interval $[c, l_i]$ (if $x = c$ $\Rightarrow$ $f_i(x) = 1$, we have the best cut, if $x = l_i$ $\Rightarrow$ $f_i(x) = 0$, we have the worst one), and
\[ r_i(x) = |1 - \frac{(x - c)}{(l_i - c)}|^n \]

denotes a factor which can alter the behaviour of the function depending on the degree of layout-standardization of the underlying document class: the more \( n \) is high, the more \( r_i(x) < 1 \) and the function \( v(x) \) is selective, being high the degree of standardization of the underlying document class. Therefore, growing \( n \), in order to have \( v(x) \) close to 1 a sharper cut is needed.

In [9] when the system is not able to classify a given document, it provides a knowledge acquisition component which allows the modification and extension of the knowledge base, interacting with the user who completes the layout description by setting cuts and labels. The result is a new physical class.

Similarly in [10] is presented a method to recognize the layout structures of multi-kinds of table-form document images, and to construct a model for each type of table-form.

The system is organized in a \textit{Structured Description Tree}, which describes a logical class of table-forms, and in a \textit{Classification Tree}, which contains different physical models that can be described by the same logical one. The Structured Description Tree consists of a \textit{Global Structure Tree} and a \textit{Local Structure Tree}. In the \textit{Global Structure Tree} meaningful sets of adjacent objects (here called item fields) which are not characterized by a complete dependent relationship among them (i.e. which have not vertical or horizontal complete adjacent relationship one another) are described. A block of rectangles or more than one vertically or horizontally repeated blocks are represented by a node of the \textit{Global Structure Tree}. The construction of such a tree is performed by scanning the document from top to bottom. Then a method able to identify the cross-points among vertical and horizontal lines is adopted; the item fields are replaced by left-upper corners. Each local structure tree is attached to the corresponding node in the global structure tree, which describes the internal structure of each block, represented by a set of item whose rectangles are not characterized by complete dependent relationship. The item fields that can not be further divided are terminal nodes, intermediate nodes represent the type of connection (vertical or horizontal) to other item fields or blocks.

In [10] a document class is defined as a set of table-form documents whose layout structures can be uniquely identified by the same layout knowledge. Item fields for printed characters are distinguished as name fields and other item fields are as data fields. In the structure extraction phase, the dependent relationships between name fields and data fields are extracted, using the generalized composition rule, based on horizontal and vertical dependent relationships. The layout and logic structure of the table-form documents doesn’t induce any ambiguity of semantic attribution.
2.3. TYPES OF ARCHITECTURES FOR DOCUMENT PROCESSING

The knowledge about document classes must be able to categorize many different table-form documents effectively so that appropriate knowledge of layout structures can be applied to individual documents of the distinguished document classes. The knowledge of layout structures is represented with a multi-way tree, it is called Classification Tree. Each node corresponds to a document physical class. The parent nodes represent a wider class of documents with respect to its children. To each node of the Classification Tree a node of the Structure Description Tree is associated. It describes the related logical structure of the correspondent logical class. Otherwise, a node of the Structure Description Tree can be associated to more than one node of the Classification Tree, because such layout structures can be described by the same logical one, thus they belong to the same logical class. The Classification Tree makes it easy to analyze the physical characteristics of layout structures because it depends on only the number and length of line segments in this specific domain.

The layout knowledge is acquired during the document analysis. Then the document physical class is identified using the Classification Tree (document classification). If such physical layout is individuated, its logical one in the Structure Description Tree is associated to it (document classification), otherwise the new physical layout, which represents a new physical class, is added to the Classification Tree. It can be a child of an existing node, thus representing a physical sub-class or a new node linked to the root that defines the empty document.

The presence of such logical structure is verified in the Structure Description Tree, otherwise the tree is updated: through the new physical structure a new logical one is build up by codified rules, and knowing a priori objects which containing data or name of the data of the correspondent object. Finally the tree which describes the logical structure of a table is linked to its physical layout.

Basically, this system purposes to recognize a physical structure and to associate it with its logical representation (document understanding). In the event that the physical model of a table-form is not contained in the knowledge base, the system constructs the new model for the current physical class, acquiring its physical and logical features, and adds it monotonically in the database.

In [11] a system for document class representation and interpretation is presented. The whole world of documents is divided into classes and a knowledge, represented by a model of layout for a specific document class is used during the understanding phase in order to extract the enclosed information. The models contain qualitative spatial relationships between objects as far as vertical and horizontal directions is concerned. The position alternatives between two objects are provided by a probability value. Therefore, in the understanding phase objects are located by a probabilistic labelling procedure.

The main goal of [11] is to improve the process of spatial document interpretation, aim-
ing at an automatic acquisition of the knowledge on document classes and semi-automatic generation of models from training data. During the training phase, the qualitative spatial models are derived from representations of the labelled training documents. Such representations are used for the logic labelling by matching an instance of document against a model.

In [11], layout parts, called objects, \((O = \{o_1, ..., o_n\})\) are organized in a tree where there is exactly one layout object with type label PAGE representing the root of the tree. The sons of the page object are block objects, their sons are line objects and so on.

The function \(p: O \rightarrow O\) maps a layout object to its parent. The set of layout objects is translated into an attributed directed graph, which is a 4-tuple \(G = (V, E, \mu, \nu)\), where \(V\) is the set of vertices, \(E \subseteq V \times V\) is the set of edges, \(\mu: V \rightarrow A_V\) maps vertices to the vertex attributes and \(\nu: E \rightarrow A_E\) maps edges to the edge attributes. The set of vertices \(V = \{v_1, ..., v_{|V|}\}\) contains exactly one element for each layout object of the document. The set of edges \(E = \{e_{ij} \mid 1 \leq i, j \leq |V| \land i \neq j\}\) contains exactly one edge \(e_{ij}\) between two vertices \(v_i, v_j\) if \(o_i, o_j : o_i = p(o_j) \lor p(o_i) = p(o_j)\). Hence, the edges describe relative locations between layout objects and their sons, and between layout objects having the same father.

The vertex attributes are 2-tuples \(A_V = (l, c)\), where \(l \in L\) denotes the layout type and \(c\) holds the logic label of the layout object. For training purposes this label must be specified manually. Each edge attribute is a 2-tuple \(A_E = (h, v)\) describing qualitatively the relative location between the vertices \(v_i\) and \(v_j\), one for the horizontal and the other for the vertical relationship.

The basis for the qualitative spatial relations is Allen’s interval algebra consisting of 13 relations \(\mathcal{A} = \{<, m, o, fi, di, s, =, si, d, f, oi, mi, >\}\) that can be used to express any qualitative relation between two intervals. For the representation of the horizontal and vertical components a 13-dimensional vector is used; each dimension represents one qualitative relation: given a relation \(a \in \mathcal{A}\) the components of \(r = (r_0, ..., r_{12})\) are calculated as follows:

\[
    r_i = \begin{cases} 
    1.0 & \text{for } i = a \\
    0.0 & \text{otherwise} 
    \end{cases} \quad (2.15)
\]

Having defined this qualitative spatial description the next step is to use it for training models automatically. The representation of single documents as attributed directed graphs forms the basis for the knowledge representation of a specific class of documents. As well as for single documents, the representation for the classes of documents is realized as an attributed direct graph and it is called class model \(M = (V_M, E_M, \mu_M, \nu_M)\). Aim of the learning process is to create a class model \(M\) from a set of labelled documents with a minimum amount of interaction. Having created a training set of labelled documents, the learning procedure runs without any further interaction. The algorithm starts with a
2.3. TYPES OF ARCHITECTURES FOR DOCUMENT PROCESSING

generic class model $M_0 = (0, 0, \mu, \nu)$ and then updates it incrementally by each document of the training set. The first part updates the vertices: if a vertex of the training document is labelled, it is included in the class model. The second part updates the edges: if an edge connects two labelled vertices, this edge is included into the class model. A spatial relation represented by the attribute of an edge is updated by

$$
\begin{align*}
\mathbf{h}_M &= \frac{i \cdot \mathbf{h}_M + \mathbf{h}_G}{i+1} \\
\mathbf{v}_M &= \frac{i \cdot \mathbf{v}_M + \mathbf{v}_G}{i+1}
\end{align*}
$$

(2.16)

where $i \in \mathbb{N}$ is the number of times an edge is updated.

For document understanding an inference mechanism which performs spatial document interpretation on documents of a previously learned class is proposed. Both a document and a class model are represented as attributed directed graphs with the same structure of vertex and edge attributes. Graphs of documents have exactly one vertex for each layout object while graphs of a class model have one vertex for each learned logical object. The inference mechanism is realized as a search for an error tolerant subgraph isomorphism. The result is a probabilistic semantic labelling of objects of a known class.
Chapter 3

The design of a multi-class document processing architecture

3.1 Introduction

The problem of designing an automatic reading system that can be used for processing as wide typologies of document as possible, faces with the variability of document physical and logical structure, both from a domain to another both inside the same domain.

In § 2.3 a number of works which reflects the most widely used architectures for document processing have been described. Particularly, the works which can be found in literature have been divided into three classes: Closed, Open and Semi-open architectures. Open and Closed architectures towards document processing show particular characteristics and properties, which make each one more suited than the other for particular requirements of the project.

Systems oriented to understand a single physical or logical class of documents (§ 2.2), namely Closed architectures, use models that contain physical object coordinates [1], [2], or logical relationships among objects [4], [3]. They usually exhibit good performances in object locating, because of the invariability of physical or logical structures of the documents they are designed to process. On the contrary, they lack flexibility, since they do not provide the possibility of processing other different layouts. The only possibility is concerned with changing the model by user interaction, as described in [1] and in [2].

On the other hand, systems oriented to read multi-classes or multi-domains of document images (Open architectures) show a considerable flexibility, but a certain weakness in the understanding phase since they are concerned with a probabilistic semantic labelling of physical objects, with loss of reliability, which depends on the variability of the classes of the domain ([5], [17], [6], [7]). Particularly, in such architectures the lack of the document classification, preliminary to the understanding phase, increases the uncertainty of the
understanding phase itself, since it forces the system to associate a unique general knowledge to different physical or logical classes. Such general knowledge describes the domain which is applied to in terms of physical constraints or positional relationships among the objects of the domain documents. Therefore, because of the variability of the structure of the classes of the domain, the use of such a knowledge leads to a probabilistic semantic attribution to the extracted physical objects during the understanding phase. The result is a loss of reliability, balanced by an increasing flexibility.

Semi-open architecture systems are generally used to develop a unique system able to process multi-classes of document images. They have been proposed in order to come to a compromise between the flexibility of open system architectures and the considerable performances of systems oriented to the understanding of a physical or logical (§ 2.2) class of documents. In order to obtain these requirements such kinds of architecture ([9], [10], [12], [13]) present a variety of models to be applied to each class of the domain of interest. Moreover they provide an intermediate level of processing between document analysis and document understanding, the document classification, which is intended to associate, to a recognized class, its specific model. The presence of the phase of document classification between document analysis and document understanding represents the specific characteristic of Semi-open architectures.

Document classification allows to greatly reduce the variability of the relationships between layout and logical structures which a document of a class may present [13], with respect to the variability of the logical-layout relationships allowable for a document in the whole domain. If a specific model can be defined for each class, document classification allows to associate a particular model to each class. Therefore, the use of such a specific model for a document of a class increases the reliability of the understanding phase with respect to the understanding phase in Open architectures. Open architectures in fact use a general knowledge on the entire domain, without recognizing the class of the current document.

Also this kind of architecture may present a certain lack of reliability. For example in [12], [13], in spite of the presence of document classification, since the classes have a certain variability of logical or physical structure, the result is a system provided with a knowledge for each class which leads to a probabilistic semantic labelling of the extracted physical objects. The degree of reliability of the understanding phase in Semi-open architectures reaches the degree of reliability of the understanding phase in Closed architectures if the domain of interest can be partitioned in physical or logical classes in the sense of § 2.2. In this case, in fact, the model of a class contains the deterministic constraints which are sufficient to univocally locate the objects, as well as in Closed architectures.

Moreover, the use of a module of document classification introduces a certain flexibility in the domain of interest, because it allows to use a single system to process different classes of documents. This is a desirable property for a document processing system, which is intended to enlarge the goal of reading of Closed architecture systems.
The aim of the present work is to design a general framework that is able to process documents belonging to a multi-class domain, with constraints of flexibility within the domain and reliability, as far as understanding phase is concerned.

For the discussed properties, Semi-open architectures seems to be the most adequate architecture for a system when documents of a multi-class domain are to be processed with constraints of reliability and flexibility.

In particular, in the present work, the term “flexibility” is associated to two different meanings: it can denote the system capability of reading documents belonging to different known classes, or the system capability of processing documents of the domain whose model is not known yet.

As discussed, in literature the flexibility inside a whole of known classes is usually solved by a procedure of document classification, which is intended to recognize the class of the current instance and to map the related model to the class [8], [9], [10], [13].

The flexibility outside a whole of known classes requires the construction of a new model for an unknown class. In literature this kind of problem is usually solved by human interaction. For example in [8], [9], the construction of a new model requires that the user himself indicates the position and the semantic of each object for the new model of the unknown class. In [12], [13], [11], a new model is obtained, by machine learning techniques, from a set of training data, that are documents belonging to the same class whose objects are labelled by the user. In [10] such a problem is solved by a set of rules which are intended to extract the logical structure in terms of object relationships but without any problem in object semantic attribution, since the physical layout itself provides the deterministic rules to assign the semantic label to the extracted objects.

In the following sections a possible Semi-open architecture for a general document domain which can be divided into classes is illustrated. One of the main characteristics of such an architecture is a procedure that is intended to obtain a semi-automatic construction of a model for an unknown class of the domain of interest.

### 3.2 A Semi-open architecture for processing documents of a multi-class domain

As discussed in § 3.1, when a domain of documents can be divided into classes, in the sense defined in § 2.2, it seems natural, for a document processing system, using a Semi-open architecture scheme. In fact the possibility of constructing or simply associating a particular model to a particular class, as well as recognizing the class of a document instance, guarantees the system to maintain the high performances in the reading task of Closed architectures and the flexibility of the Open architectures within a particular domain of interest.
As it has been described in § 2.3 and in § 3.1, a *Semi-open architecture* is characterized by the possibility of classifying an instance in a known class and using its model for the phase of *document understanding*, or constructing a model for each class of documents the domain of interest consists of.

Our approach presents a general framework for processing multi-class document domain. In particular, it is considered the subdivision of a domain in *physical classes* (§ 2.2), which are characterized by the position invariability among the objects in terms of (absolute or relative) coordinates, but, similarly, a *Semi-open architecture* system can be designed for *logical classes* (§ 2.2) characterized by the qualitative position invariability among the objects.

The system is designed on a knowledge-based approach to document processing. It is based on specific document class models for the understanding phase of known classes; such models are part of a general knowledge, which describes the domain of interest as far as the physical, the logical structure and their mutual relationships is concerned, trying to capture the similarities among the classes. Such a general knowledge is used both as a procedural scheme for the phases of *document analysis, classification and understanding* of known classes, both as a tool for the construction of *Document Models* for unknown classes.

### 3.2.1 The Document Model

If the document domain of interest is composed by classes characterized by a fixed logical and physical structure, namely physical classes (2.2), the logical-layout relationships of documents belonging to a class can be described by a specific model that we call *Document Model*.

A *Document Model* $\mathcal{DM}$ of a physical class of documents, consisting of $n$ objects $\{d_0, d_1, ..., d_{n-1}\}$, is the set of $n$ 5-tuple $d_i = (l_i, x_{i_{\text{min}}}, y_{i_{\text{min}}}, x_{i_{\text{max}}}, y_{i_{\text{max}}})$, $0 \leq i \leq n - 1$, where $l_i$ is the semantic label of $d_i$ and $(x_{i_{\text{min}}}, y_{i_{\text{min}}}, x_{i_{\text{max}}}, y_{i_{\text{max}}})$ are the physical coordinates which locate univocally $d_i$.

Shortly, we can indicate an object $d_i$ as:

$$d_i = (l_i, X_i)$$  \hspace{1cm} (3.1)

The *Document Model* may be *absolute* or *relative*. In the former case, each object has absolute coordinates with respect to the scanner coordinate system. In the latter, each object has relative coordinates with respect to an object $d_k \in D(n)$ considered as reference. Usually a *Document Model* with relative coordinates allows the system to be more tolerant to skew variation.

An example of *Document Model* is reported in Fig. 3.1.

Similarly, if the domain of interest can be partitioned in logical classes (§ 2.2), simply a model which contains logical relationships among objects (see [4]) could be used during
3.2. THE SEMI-OPEN ARCHITECTURE

Figure 3.1: Example of a document and the related Document Model.

the understanding phase of an instance of such a class. Thus, in this architecture, it could be sufficient to construct this kind of model for each logical class of the domain.

In such a kind of architecture a Document Model DM of a logical class of documents, consisting of objects \{d_0, d_1, ..., d_{n-1}\}, is the set of object mutual positional relationships (“left-of”, “right-of”, “above”, and so on) and eventually object physical constraints which locate univocally objects \(d_i\), and the set of semantic labels \{l_0, l_1, ..., l_{n-1}\} related to such objects.

3.2.2 The structure of the system

After having acquired a document instance by a scanner, and converting it into an electronic format, the system performs a number of low-level processing procedures which are intended to extract primitives, that are connected components [16], which specialize into segments and not segments components.

Starting from such types of primitives, it is possible to individuate 3 classes of physical objects to be assigned with a semantic, they are:

- isolated segments;
- isolated components;
CHAPTER 3. A MULTI-CLASS DOCUMENT PROCESSING ARCHITECTURE

- item rectangles;

The set of these three types of physical objects represents the layout structure of the document. “Item rectangles” represent rectangles which contain information, and can specialize into two further types:

- item rectangles surrounded by segments (segment item rectangles);
- item rectangles as a group of components (component item rectangles);

The types of physical objects to be extracted during the document analysis phase are illustrated in Fig. 3.2.

![Diagram of physical objects]

Figure 3.2: The types of physical objects to be extracted.

After having extracted the layout structure of the current document a deskewing procedure to align the coordinate system of the scanner to that of the current instance can be performed.

On their basis the system may attempts to classify the domain if it is required (minimally to verify whether the instance belongs to the domain of interest) or rejects the instance itself. In this framework such a level is only indicated as a possible procedural step, because the problem of domain classification here is not considered. Thus, it is assumed that the system is fed only with documents belonging to the domain of interest so
that the system could distinguish among the physical (or logical) classes or, at least, could
reject an instance of a different domain. In case of successful verification of the domain the
system attempts to classify the instance in a domain class. The successful classification
of the instance means having recognized the class the current document instance belongs
to. On the contrary, if the class is not recognized, the system cannot map the document
instance to its model. In this case a new Document Model is built.

Therefore, depending on the successful or unsuccessful classification, the system pro-
vides two different alternatives:

1. Instance matching with the Document Model of the class in order to assign the
semantic to each rectangle and consequently to understand the current document
instance.

Since the current instance is assigned to a particular Document Model $\mathcal{DM} = \{(l_0, X_0), ..., (l_{n-1}, X_{n-1})\}$, for each extracted object $d^*_j$ results (3.2):

$$\forall d^*_j \exists! d_i \in \mathcal{DM} : ||X^*_j - X_i||_\infty \leq \epsilon$$

where $\epsilon$ is an upper bound threshold.

Therefore the extracted object $d^*_j$ is labelled with the semantic associated to the
object $d_i$:

$$d^*_j = (i, X^*_j);$$

2. The instance does not belong to a known class of the domain, then the system rejects
the instance that can be considered the prototype of a new class. For such a new
class a new Document Model is constructed.

The result is a Document Model for the class of documents which the current instance
belongs to. Such a model can be used for the document understanding, as described
in 1).

The logical scheme of the whole system fed only with documents belonging to the
domain of interest is reported in Fig. 3.3.

It is a typical scheme of a Semi-open architecture (see Fig. 2.9); the main differences
are related to the semi-automatic construction of a new model and to the role of the user,
who has not to explicitly provides labels to the extracted objects but only has to limit his
interaction to only confirmation or refutation of a semantic label provided by the system,
as it will be specified in § 4. Moreover, in most cases, such an approach spears the system
to locate and read the keywords that indicate the semantic to the objects.
As discussed document classification is a decisive function for a Semi-open architecture in order to associate a specific model to the current instance. The role of document classification in a document processing system has already been discussed in § 2 and in § 3.2.

In § 2.2 a domain of document is identified by its subject. This kind of definition doesn’t provide an operative procedure to reliably perform a domain classification, but it is the most correct identification of a domain and the most widely used in literature. For this reason it not easy to find a general method to obtain a reliable domain classification. For a limited number of domains some features, based on the physical structure or on the logical one can be individuated, so that the procedure of domain classification can be performed.

Usually it can be based on the detection of particular graphical elements or structural sequences of elements which are specific of a particular domain.

Our framework is described for a domain that can be divided into classes, thus the procedure of domain identification can be neglected or can be restricted to a procedure of verification whether an instance belongs to the domain of interest or has to be rejected.

The method of classification of documents within a particular domain is strictly bounded to the domain of interest, to the layout characteristics of the documents and to the equivalence relation that divides the domain in classes. Some domains show particular graphic elements, for example logo, segments, geometrical objects which identify univocally the physical or logical class, in the sense of § 2.2.

The physical structure itself combined with the identification of particular graphic elements or with the identification of specific keywords of an instance can be used to identify a class of the domain of interest. Being

\[ d_j^* = (\text{Null}, X_j), \quad j = 0, \ldots, n - 1 \]  

(3.4)
3.2. THE SEMI-OPEN ARCHITECTURE

the set of $n$ objects extracted from an instance, and

$$X_j^* = [x_{j_{\text{min}}}^*, x_{j_{\text{max}}}^*, y_{j_{\text{min}}}^*, y_{j_{\text{max}}}^*], \ j = 0, ..., n - 1 \ (3.5)$$

doing the coordinates of the object $d_j^*$ of the current document instance, if

$$\forall d_j^* \exists! d_i \in D\mathcal{M}_N : ||X_j^* - X_i||_{\infty} \leq \epsilon \ (3.6)$$

the instance of document corresponds to the layout of the $N^{th}$ class. Eventually, the layout matching can be integrated by the use of an OCR system to verify the semantics of each object, or a module of graphical elements identification, to avoid as less as possible the to associate to the same class documents with the same layout structure but different semantics of the physical objects. If all this condition are satisfied the current instance can be assigned to the $N^{th}$ class and the related Document Model can be used to label each extracted object.

As discussed, since the problem of document classification depends on the layout characteristics of the domain of interest, in the present work it is not specifically treated, but simply as a procedural part of the document processing procedure (see 4.3.2).

3.2.3 The Conceptual Model

The entire system can be based on a general knowledge related to the domain of interest which is intended to describe the document domain at physical and logical-layout levels, trying to capture some features common to the various classes which the domain consists of. Moreover, it contains the specific knowledge for the known classes of documents of the domain, which is represented by the Document Models. Such a knowledge is called Conceptual Model.

The Conceptual Model for a specific domain of documents is a declarative knowledge base, described by frames of a semantic network. The model, its semantics and its pragmatics are described in § 4.

The knowledge provided by the Conceptual Model describes the domain of interest under two different aspects.

The first one provides a general description of the layout structure of the documents of the domain: it describes the physical objects a document of the domain may consist of (for example connected components, segments, rectangles). Moreover, it contains the specific description of each known class of the domain in terms of Document Models.

The second one provides a general description of the logical-layout relationships among the objects of the domain. Therefore it represents a general model that describes the physical constraints of each logical object, which may be present in a document of the domain, and their positional relationships with respect to other logical objects.
CHAPTER 3. A MULTI-CLASS DOCUMENT PROCESSING ARCHITECTURE

Using the general framework of the Conceptual Model for a domain of interest, every phase of document processing (document analysis, classification, understanding and Document Model construction) can be performed.

and

More details about the use of the Conceptual Model to perform document processing will be discussed in § 4. In this paragraph, with reference to Fig. 3.4, only the parts of the Conceptual Model used for each single phase of document processing and their interactions are discussed.

When an instance of document is acquired the part of the Conceptual Model that declares the physical characteristics of the domain can be used in order to extract graphical primitives and, on their basis, the physical objects. Therefore the phase of document analysis is performed. The output of the document analysis phase is represented by the layout structure (§ 2.1) of the current instance of document (Fig. 3.4). The layout structure represents the collection of the extracted physical objects. On the basis of the layout structure, the phases of document classification, document understanding and Document Model construction can be performed.

In fact, if the layout structure is acquired, then the phase of document classification, eventually supported by an OCR system, can be executed. Such a phase involves both the layout structure both the set of Document Models of known classes which the current instance may be associated to, in case of successful classification of the instance. Therefore, the layout structure represents the intersection between the phase of document analysis and document classification (Fig. 3.4). In case the class of the document is recognized, the phase of Document Model construction, followed by the phase of document understanding that uses the constructed Document Model, can be performed. The phase of Document Model construction involves the two aspects of description of the domain: the logical-layout description and also the physical description in terms of the layout structure of the current instance and in terms of general description of the layout structure of the domain. Therefore the phase of Document Model construction contains the document analysis phase and has a not empty intersection with the document understanding, represented by the layout structure, and a not empty intersection with the document classification phase, represented by the layout structure and the new Document Model (Fig. 3.4).

Both in case the class is recognized both it is not recognized and a new Document Model is to be constructed, the document understanding phase contains the document classification phase: in fact, classifying a document, in the context of this system, means having associated the related Document Model to the current instance, namely having understood the document.

Therefore, the use of such a knowledge for a particular domain of documents provides the system with a procedural scheme for processing documents of known classes, and drive the system to construct a new Document Model for an unknown class of the domain of documents the Conceptual Model refers to.
These characteristics and use of the Conceptual Model ensures flexibility to the system in the document domain of interest.

The phase of Document Model construction is a semi-automatic procedure, since it provides user interaction but only for confirmation or refutation of a semantic attribution suggested by the user, as it will be specified in 4. The output of such a procedure is a Document Model for each unknown class.

In this kind of architecture the procedure of Document Model construction represents a critical point to ensure flexibility and reliability to the whole system. Each Document Model is associated to the related physical class of the domain. The Document Model of a class can be used directly for the reading task if the class of the current document instance is recognized.

Globally, the parts of the Conceptual Model and their interactions are described in Fig. 3.4.

Figure 3.4: The parts of the Conceptual Model and the interaction scheme among them.
Chapter 4

A Conceptual Model based on a semantic network

As discussed in § 3, the proposed framework consists of a knowledge-based approach: a Document Model (DM) for each class of the domain, which is used for the understanding phase of known and recognized classes, and a general knowledge, which the Document Models are part of, that is used for the semi-automatic construction of a new Document Model for an unknown class of the domain.

Multi-class document domain, in fact, is usually characterized by a wide variety of physical or logical classes, in the sense of § 2.2. Nevertheless, it is usually possible to describe the domain by a general knowledge, oriented to capture some similarities among the document classes of the domain. Such similarities could be used directly for the understanding phase, but they would lead to a probabilistic semantic attribution to the extracted physical objects. This fact would greatly reduce the reliability of the system, while the goal is to keep a high reliability increasing the flexibility inside a domain of interest at least.

In this framework such a knowledge describes the document domain of interest by a semantic network, inspired on [28], where the objects of the domain are described at physical and logical level. Such a knowledge has been called the Conceptual Model (CM).

4.1 The syntax and the semantics of the network

The semantic network we describe allows us to represent an a priori knowledge on the document domain of interest by nodes and directed labelled edges, called links. Each node represents a concept or a class of concepts, or is a description of individuals. A node is described by a set of characteristics, some of them are further specified by particular substructures. Edges are used to express binary relations between the nodes. Moreover
such network allows to describe the document domain by different levels of abstraction: each level focuses on particular logical or physical characteristics; a specific link to pass from a level to another one is provided.

<table>
<thead>
<tr>
<th>GENERAL_TYPES:</th>
<th>inverse links/</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>specialization of</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NODE_TYPES:</th>
<th>part of</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept</td>
<td>concrete of</td>
</tr>
<tr>
<td>instance</td>
<td>instance of</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>link</th>
<th>substructure</th>
</tr>
</thead>
<tbody>
<tr>
<td>specialization</td>
<td>attribute description</td>
</tr>
<tr>
<td>part</td>
<td>link description</td>
</tr>
<tr>
<td>concrete</td>
<td>function description</td>
</tr>
<tr>
<td>instance</td>
<td></td>
</tr>
</tbody>
</table>

A concept node represents classes of objects, having certain common properties. The special case that a concept represents only one element is not excluded. An instance node on the contrary represents a particular manifestation of a member of a concept. A concept and an instance of it are related by an instance link. Together with the links specialization and part, it represents a hierarchies, that define a partial non-reflexive order on the set of nodes called axis. The axes defined by the links instance, specialization and part are called respectively:

- Classification: type of hierarchy that forces a distinction between a concept and an instance which is an extensional description of a concept. The related link is called instance of and its inverse instance;
- Generalization: type of hierarchy that relates a concept to more generic ones. The related link is called specialization of and its inverse specialization; in other frameworks is often called is_a;
- Aggregation: type of hierarchy that connects a concept with other concepts which describe its parts. The related link is called part and its inverse part of. In this context, aggregation models a physical part hierarchy [29].

Figure 4.1: General data types in the syntactic definition of the network structure.
The link concrete establishes relationships between different levels of abstraction and a hierarchy among them.

The nodes of the semantic network are described by frames. The data structures of the concept nodes and the instance nodes are identical, except that a pointer to a function in a concept is replaced by the computed value in the corresponding instance. Moreover, a concept node contains the slot “instance”, which is linked to the related instance nodes, while the instance node contains the slot “instance of”, linked to the related concept nodes. Note that there are connections between one real world object and different concepts, as well as connections between a concept to different collections of real world are admissible.

The general structure of a frame that describes any concept of the semantic network is reported in Fig. 4.2, where the layout of each substructure, indicated within square brackets, linked to a slot, is indented below the slot itself.

![Figure 4.2: A general frame for a concept of the semantic network.](image)

Each type of concept frame contains the following slots (see Fig. 4.2):

- “Name of concept”;

- “Frequency”, which indicates the observed frequency of the object modelled by such a concept;

- “Attribute”, which is described by a particular substructure, “attribute description”, in terms of “Role”, “Type of values”, “Number of values”, “Restrictions”, “Meanings”, “Values” and “Computation of values”. The slot “Computation of values” contains a pointer to the substructure “function description”, which describes the function computing the value of the related attribute; its specification slots are “Name” and “Arguments”. If the attribute is represented by a state variable, the
slot “Meanings” contains an array with the names of the elements of the variable, and the slot “Values” the array of the probability values of such states.

- “Specialization”, which contains a list of links to other nodes that specialize the related concept;

- “Part” and “Concrete”, which contain a list of links to the substructure “link description”. This substructure describes the role of a part (or of a concrete), by the slot “Role”. Each concept which is part or concrete of the current concept is attached to the slot “Goal Node”. Finally, the slot “Frequency” provides the observed frequency of such a part (or concrete);

- “Judgement”, which contains a link to the substructure “function description”. Such substructure contains the slots “Name”, which indicates the name of the function, and “Arguments” which contains a list of arguments for the function. The function, attached to such a substructure, computes a judgement of an instance, therefore it is responsible of the instantiation of its concept.

- “Instance”, which contains a list of instances of the related concept.

The introduced network is used to describe the Conceptual Model of a multi-class domain of documents and it allows to describe the document domain at four different levels of abstraction (Fig. 4.14):

```
image level – primitive level – geometry level – named object level
```

The “image level”, in practice, refers to the pixel level; the “primitive level” is concerned with the treatment of primitive objects as connected components and their specializations as segments and other graphical objects; the “geometry level” describes the geometric objects obtained by segments and by the clustering of the connected components; the “named object level” is concerned with the description of the whole domain at semantic level and the related physical and contextual constraints.

4.2 A Conceptual Model for a multi-class domain of documents

The semantic network, introduced in § 4.1, is used to describe the Conceptual Model for a multi-class domain of documents (Fig. 4.14 can be considered as reference).

At the “named object” abstraction level, the concept DOC_OF_DOMAIN is the root.
4.2. A CM FOR A MULTI-CLASS DOMAIN OF DOCUMENTS

<table>
<thead>
<tr>
<th>Part: Concept_0</th>
<th>Attribute: height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal_Node: Cpt_0</td>
<td>Type_of_val: INTEGER</td>
</tr>
<tr>
<td>Frequency: f_0</td>
<td>Numb_of_val: 1</td>
</tr>
<tr>
<td>...</td>
<td>Restriction: —</td>
</tr>
<tr>
<td></td>
<td>Comp_of_val: comp_document_height</td>
</tr>
<tr>
<td>Part: Concept_k</td>
<td>Attribute: width</td>
</tr>
<tr>
<td>Goal_Node: Cpt_k</td>
<td>Type_of_val: INTEGER</td>
</tr>
<tr>
<td>Frequency: f_k</td>
<td>Numb_of_val: 1</td>
</tr>
<tr>
<td>...</td>
<td>Restriction: —</td>
</tr>
<tr>
<td></td>
<td>Comp_of_val: comp_document_width</td>
</tr>
<tr>
<td></td>
<td>Arguments: IMAGE.height</td>
</tr>
<tr>
<td>Part: Concept_t</td>
<td>Arguments: IMAGE.width</td>
</tr>
<tr>
<td>Goal_Node: Cpt_t</td>
<td>Judgement: judge_doc_of_domain</td>
</tr>
<tr>
<td>Frequency: f_t</td>
<td>Arguments: IMAGE, LAY_STRUCT</td>
</tr>
<tr>
<td>...</td>
<td>Frequency: 100%</td>
</tr>
<tr>
<td></td>
<td>Instance: D</td>
</tr>
</tbody>
</table>

Figure 4.3: A frame description of the concept DOC_OF_DOMAIN.

A concept DOC_OF_DOMAIN consists of parts which are indicated as Cpt_0, ..., Cpt_k, ...
Moreover each of them may consist of some other parts. Cpt_0, ..., Cpt_k, ...
are typologically concepts, in the sense of the syntax of the network, moreover they are
generically called “Concepts” since they acquire the name only from the particular domain of
interest. They represent the generic objects a document of the domain described by the
semantic network may consist of. Each concept at the “named object level” is linked to
one or more concepts at the “geometry level”.

On the other side, at the “image level” a document is simply an image represented by
the concept IMAGE, namely a set of pixels which can be grouped in clusters of connected
pixels (CONN_PIXELS).

Such clusters, at the “primitive level” represent connected components (CONN_COMP),
which specialize in segments (SEGMENTS) and in components which do not form segments (N_SEGM_COMP). In their turn, not segment components specialize in components
belonging (COMP_OF_RC) or not belonging (N_OF_RC_COMP) to rectangles which contains
information, called item rectangle. Components belonging to item rectangles are part of item rectangles without segments (RC_WHOUT_SEGM), which are obtained by a clustering of connected components. Segments specialize in segments which belong or do not belong to item rectangles (N_OF_RC_SEGM, SEGMM_OF_RC). Segments belonging to item rectangles are part of RC WT_SEGM.

Segments which do not belong to an item rectangle are linked with a concrete of link to the concept ISOL_SEGM at the “geometry level”. As well, item rectangles with segments are linked to the concept SEGMM_ITEM_RC, item rectangles without segments to COMP_ITEM_RC, and components which do not belong to item rectangles are linked
to \textit{ISOL\_COMP}.

Still at the “geometry level”, \textit{SEG\_ITEM\_RC} and \textit{COMP\_ITEM\_RC} are specialization of item rectangles represented by the concept \textit{ITEM\_RECT}. Moreover \textit{ITEM\_RECT}, \textit{ISOL\_COMP} and \textit{ISOL\_SEG\_M} are part of the layout structure which is represented by the concept \textit{LAY\_STRUCT} described in Fig. 4.4.

| LAY\_STRUCT | Specialization: class of documents |
| Part: isolated segments | Goal\_Node: ISOL\_SEG\_M |
| | Frequency: 100% |
| Part: item rectangles | Goal\_Node: ITEM\_RECT |
| | Frequency: 100% |
| Part: isolated components | Goal\_Node: ISOL\_COMP |
| | Frequency: 100% |

Figure 4.4: \textit{A frame description of the concept LAY\_STRUCT.}

As discussed in § 3.2.3 the layout structure of a document image represents the collection of all the extracted physical objects, which can be grouped in isolated segments, isolated components and rectangles which contain information (item rectangles).

The part of the domain described so far, that is the part from the “image level” to the concept \textit{LAY\_STRUCT} of the “geometry level”, is intended to provide a procedural scheme for the document analysis phase.

Hereinafter, the description of the domain at the “geometry level” is functional to the document classification and the document understanding phase or to the construction of a new Document Model for an unknown class. In fact, in the part used for document classification and document understanding, the layout structure (\textit{LAY\_STRUCT}) specializes in a number of concepts \textit{DOC\_CLASS\_p}, equal to the known class of the domain so far; \( p \) is the order number of the class. Each \textit{DOC\_CLASS\_p} consists of parts which are the objects the particular \( p^{th} \) known class is composed of. They are represented by the concepts \textit{OBJ\_p}, where \( i \) is the index of a particular object of the \( p^{th} \) class. Moreover the concepts \textit{OBJ\_p} are linked to the correspondent concept \textit{Cpt\_j} at the “name object level” with a concrete of link.

In the part used for the Document Model construction, the item rectangles (\textit{ITEM\_RECT}) specialize in item rectangles with particular physical constraints, referred to their dimensions and indicated by a state of the arrays \textit{hcat} and \textit{wcat} of the correspondent concepts at the “named object level” (\textit{IT\_RC\_hcat(j)}, \textit{IT\_RC\_wcat(k)}).
As well, components and segments which do not belong to item rectangles \( (ISOL\_COMP, ISOL\_SEGM) \) specialize in isolated components and segments with particular physical constraints indicated by a state of the arrays \( hcat \) and \( wcat \) of the correspondent concepts at the “named object level” \( (ISOL\_COMP_{hcat}(j), ISOL\_SEGM_{wcat}(k)) \).

Each of such concepts at the “geometry level” are attached to one or more concepts \( Cpt_i \) at the “named object level” with a concrete of link.

As an example of a frame description of a concept at the “named object level”, the general structure of a frame that describes the concept \( DOC\_OF\_DOMAIN \) is described in Fig. 4.3. The frame declares the name of the concept and the name of the concepts it is linked to. In particular the slots “Part” indicate the concepts which \( DOC\_OF\_DOMAIN \) is linked to, namely \( Cpt_0, Cpt_1, ..., Cpt_t, ... \). The slot “Frequency” of the substructure “link description”, that describes the “Part” link, declares the observed frequency of such a part concept. Moreover the slots “Attribute” describes the value of height and width of an image of a document that represents a \( DOC\_OF\_DOMAIN \). The slots “Comp_of_val” of the substructure “attribute description” contains a link to the functions that calculate the width and the height of an instance of \( DOC\_OF\_DOMAIN \), respectively. Their arguments are the values \( IMAGE\_width \) and \( IMAGE\_height \) respectively. The slot “Judgement” contains a link to the function “judge_doc_of_domain”, whose arguments are the image, represented by the concept \( IMAGE \) and its layout structure, represented by the concept \( LAY\_STRUCT \). Such a function may decide whether an acquired image is a document of the domain of interest. Finally, the slot “Instance” contains a link to a list of instances of documents of the domain.

4.3 The pragmatics of the network

As discussed in § 3.2, the Conceptual Model for knowledge representation on a multi-class domain of documents can be used to process a known or an unknown class of documents in the domain of interest. The possibility of describing a multi-class domain of documents at different abstraction levels provides a procedural scheme for document processing that comprises the three phases which a document processing system based on a Semi-open architecture consists of.

As described in § 4.2, rising the network from the “image level” (the pixel level), through the “primitive level”, till the instantiation of the concept \( LAY\_STRUCT \) at the “geometry level”, the system analyses (document analysis) the current document instance. In the hypothesis that document classification is performed on the basis of the layout structure, as it is usually carried out in literature [8], [9], [10], at this level of instantiation of the concepts of the network all the information for classification of the current document
instance is available. Therefore the procedure of document classification can be executed. Depending on the difficulty of discriminating among the classes of the domain of interest, the phase of document classification can also be supported by an OCR system which preliminarily provides the semantic to some physical objects. Then, the phase of document understanding can be performed, in the sense of associating a Document Model to the current document instance, or of constructing a new model for an unknown class, by using the relationships between the “named object” and the “geometry level”.

In particular, the procedure of Document Model construction consists of selecting each concept at the “named object level” from the Conceptual Model, searching for a number of candidate physical objects to associate to the current concept and assigning, to the object with the highest credibility, the semantic label related to the current concept. In this architecture such a procedure is performed under the Conceptual Model supervision and driven by it; user interaction is also provided only for confirmation or refutation of a labelling hypothesis suggested by the Conceptual Model. Such an interaction is justified by the fact that object semantic labelling is a “key” procedure in a Document Model construction. An error in the labelling phase of an object, in fact, spreads to the others, with the risk of making every other semantic attributions fail.

4.3.1 The Reading Plan

The selection of a concept at the “named object level” to be instantiated is driven by a Reading Plan associated to the network. The introduction of a Reading Plan is inspired on psychological studies about the process of human reading [5]. Readers allocate attention among their knowledge resources, and go back and forth from their knowledge to the data base of the text. An information processing analysis assumes that any cognitive task can be understood by analyzing it into stages that proceed in a fixed order over time, beginning with sensory input and ending with an output or response. In general feedback loops can be inserted at any point of the process of reading varying the degree of complexity of the process. The introduction of a Reading Plan responds to this theoretical assumption, but it is intended to avoid backtracking in order to reduce the complexity of the problem.

In the architecture proposed in this work, the Reading Plan represents an order among the concepts at the “named object level”. Hereinafter, when simply the term “concept” is used, a concept at the “named object level” is to be intended. The concepts are ordered with respect to their contextual dependencies, to a decreasing observed frequency and to the priority of reading. Such an order provides a priority for locating objects in the labelling phase, therefore objects are located with respect to physical and contextual constraints related to objects previously labelled. Providing a priority of reading without back-tracking, the Reading Plan is able to greatly reduce the search space at each step, enhancing the likelihood of right location of following objects.

In the following paragraphs the function “Document processing” and the main sub-
functions for the Document Model construction, which are specific of this architecture, will be described. As far as the function related to document analysis and document understanding is concerned, can be chosen among the correspondent procedures used in literature and according to the features of the domain. As discussed in § 3.2.2 the procedures for document classification are not treated here, since they do depend on the features of the domain.

4.3.2 The Document processing procedure

The whole system for processing documents of a multi-class domain can be described by the procedure “document_processing” (Algorithm 4.1).

Such a procedure is intended to performs the phases of document analysis, classification, understanding and the phase of Document Model construction. The functions that perform such phases are invoked by the procedure “document_processing”.

The phase of document analysis is intended to extract the physical objects, whose types are reported in Fig. 3.2.

Besides the low-level procedures of primitive extractions, the procedures of item rectangles surrounded by segments (“rect_surrounded_by_segments”) and of item rectangles as a group of components (“rect_as_group_of_components”) are performed.

As discussed in § 3.2.2, the phase of document classification is indicated only as a procedural step, even if some general issues are treated in § 3.2.2. However, the problem of document classification is faced for a particular domain in § 5.

Document understanding is performed through a Document Model matching by the procedure (“Document_Model_matching”), as described in § 3.2.2.

Document Model construction is performed by the procedure “model_driven_labelling”, described in § 4.3.7. It invokes the “array_of_state_prob” (§ 4.3.7), used to performs the labelling driven by the Conceptual Model, and the “keyword_based_labelling” (§ 4.3.8) used at the end of the “model_driven_labelling” for the labelling of the objects not labelled by the model driven labelling phase. Both the “model_driven_labelling” and the “keyword_based_labelling” invoke the function “concept_constraint_updating” (§ 4.3.9) which updates the Conceptual Model.

In order to describe the steps of the “document_processing” procedure, Fig. 4.14 can be considered as reference. Each list of extracted objects is represented by the correspondent concept of the semantic network.

The function “document_processing” invokes the low-level processing procedures which provide at the extraction of a list of connected components (\(C_c\)) and of segments (\(S\)). From the list of connected components, the list of components which do not form segments (\(C\)) is selected.

On their basis a “deskewing” procedure, in order to align the coordinate system of the current instance to that one of the scanner, can be performed.
The following step is concerned with the extraction of all the physical objects a document consists of. In particular the system has to extract the rectangles which contain information (item rectangles ($R$)). Segments specialize in segments which form or do not form item rectangles; as well not segment components specialize in components which form or do not form item rectangles.

On their basis, item rectangles surrounded by segments ($SR$) and of item rectangles which result from a clustering of connected components ($CR$) can be obtained. Such types of item rectangles determine the list of all kinds of unlabelled item rectangles ($R$).

The list of segments, which do not form item rectangles ($SnR$) (also called isolated segments), and the list of connected components which do not belong to an item rectangle determined by their clustering ($CnR$) (also called isolated components), are then selected.

$R$, $SnR$ and $CnR$ determine the list of the physical objects (Obj) an instance of document consists of.

When all the objects of the current document instance have been extracted, the layout structure has been identified, that is the concept $LAY\_STRUCT$ ($LY$) is instantiated. On the basis of the extracted primitives (item rectangles, isolated segments and isolated components) and, eventually, by the use of the OCR, the procedures of domain recognition and of the class of the domain identification of the current document instance can be performed.

As it has been pointed out in § 3.2.2, since this framework is designed for a multi-class domain of document, the procedure of domain identification can be neglected or can be restricted to a procedure of verification whether an instance belongs to the domain of interest or has to be rejected. Therefore the procedure “judge\_doc\_of\_domain” verifies whether the instance belongs to the domain of interest $D$ or not. In case the instance does not belong to the domain of interest it is rejected, in the other case the class recognition can be performed with the procedure “class\_recognition”. As discussed in § 3.2.2, since the phase of document classification strictly depends on the features of the domain of interest and on the equivalence relation that divides the domain in classes, in this framework it is not developed. Only some indication of the possible strategies to be used for the classification of document.

If the current instance of document belongs to a known class, for example the $p^{th}$, its layout structure is assigned, as an instance, to the concept $DOC\_CLASS_p$, thus, considering its part, $OBJ_0^p$, $OBJ_1^p$, ..., $OBJ_t^p$, ..., the Document Model related to such a class is selected. The concept $DOC\_CLASS_p$, whose frame description is reported in Fig. 4.5, is a specialization of the concept $LAY\_STRUCT$. The slots “Part” indicate the concepts which $DOC\_CLASS_p$ is linked to, namely $OBJ_0^p$, $OBJ_1^p$, ..., $OBJ_t^p$, ..., the slot “Frequency” of the substructure “link description”, that describes the “Part” link, declares the observed frequency of such a part concepts. The slot “Judgement” contains a link to the function “class\_p\_recognition” which is intended to verify if the extracted layout structure ($LAY\_STRUCT$ is the argument of the function) belongs to the $p^{th}$ class. The slot
“Frequency” of the frame indicates the observed frequency of the concept $DOC\_CLASS_p$. Finally, the slot “Instance” contains a link to a list of documents of the $p^{th}$ class.

After the phase of document classification (“class_recognition” procedure), if the class is recognized, the procedure “Document Model matching” is intended to instantiate all the concepts $OBJ_p^i, i \in N$, where $OBJ_p^i$ is the $i^{th}$ object of the $p^{th}$ class of documents and therefore it is part of $DOC\_CLASS_p$. The concept $OBJ_p^i$, whose frame description is reported in Fig. 4.6, is in a part-of relation with the concept $DOC\_CLASS_p$ and it is in a concrete-of relation with the concept $Cpt_i$, which corresponds to its semantic label. Four slots “Attribute” indicate the four coordinates which locate the $OBJ_p^i$; each substructure “attribute description” declares the type of the coordinate value (“$Type\_of\_val$”), the number of value for each attribute (“$Numb\_of\_val$”) and the coordinate restrictions (“Restriction”). Moreover such substructure contains, in the slot “$Comp\_of\_val$”, a link to a function that calculates the value of a coordinate of the object. The arguments of the function “$Comp\_of\_val$”, for each coordinate attribute, are the list of instances of the concept $ITEM\_RECT$ and the restrictions of the coordinate itself. Such coordinate restrictions represent the tolerance, allowed by the Document Model of the $p^{th}$ class, for the correspondent coordinate in an instance of the concept $OBJ_p^i$.

<table>
<thead>
<tr>
<th>DOC_CLASS_p</th>
<th>Part: object</th>
<th>Goal_Node: $OBJ_p^{n-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specialization-of: LAY_STRUCT</td>
<td>Frequency: 100%</td>
<td>Frequency: [number]</td>
</tr>
<tr>
<td>Part: object</td>
<td>Judgement: class_p_recognition</td>
<td>Instance: [list of instances]</td>
</tr>
<tr>
<td>Goal_Node: $OBJ_p^i$</td>
<td>arguments: LAY_STRUCT</td>
<td></td>
</tr>
<tr>
<td>Frequency: 100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal_Node: $OBJ_p^{i-1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency: 100%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.5: The frame description for the concept $DOC\_CLASS_p$.

Moreover each concept $OBJ_p^i$ is linked at the “named object level” to the concept related to its label with a concrete of link.

The instantiation of the concepts $OBJ_p^i$ is obtained by matching all the extracted objects $Obj$ with such concepts, $i \in N$, obtaining a list of labelled objects ($\overline{Obj}$). Each labelled object is attached to the slot instance of the related $OBJ_p$ concept.

If the current document instance does not belong to a known class a new Document Model ($DM$) is constructed by the “model\_driven\_labelling” procedure, that is responsible of the semantic labelling of the objects of the list $Obj$ for a document of an unknown class. Such a procedure will be described in § 4.3.7.

Finally a new concept $DOC\_CLASS\_new$ for the new class is created and it is added to the Conceptual Model at the “geometry level”. The new concept $DOC\_CLASS\_new$ is attached by a specialization of link to the concept $LAY\_STRUCT$, and the concepts
Figure 4.6: The frame description for the concept OBJ\textsuperscript{p}.

OBJ\textsuperscript{i,new} related to its extracted and labelled objects are linked by a part of link to the concept DOC\_CLASS\_new, therefore the Document Model of the new class is created.

Each object of the new class can now be linked to the concept at the “named object level”, related to its label, with a concrete of link. The instances are attached to the slot “instance” of the related new concept OBJ\textsuperscript{p,new} with an instance link.

When all the objects of the current document instance are labelled, from each item rectangle which is not a graphic rectangle, words (W\textsubscript{d}) and rows of words (R\textsubscript{w}) are selected by the functions “components_to_words” and “words_to_rows”, and the textual information T\textsubscript{x} can be extracted by an OCR system.

Algorithm 4.1 DOCUMENT\_PROCESSING(image)

Input: the image of a document instance;

Output: a list of labelled objects OBJ\textsubscript{i}, and the textual information from item rectangles;

1 \hspace{1em} \mathcal{C}c \leftarrow \text{connected\_components (image)};
4.3. THE PRAGMATICS OF THE NETWORK

2 \( S \leftarrow \text{segment\_extraction}\ (Cc) \);
3 \( C \leftarrow Cc - S \);
   \(\triangleright C \text{ is the list of not segment components}\)
4 \((C, S) \leftarrow \text{deskewing}\ (C, S)\);
5 \(SR \leftarrow \text{rectangles\_surrounded\_by\_segments}\ (S)\);
6 \(CR \leftarrow \text{rectangles\_as\_groups\_of\_components}\ (C)\);
7 \(R \leftarrow SR \cup CR\); \(\triangleright\text{List of all kinds of unlabelled item rectangles}\)
8 \(SnR \leftarrow \text{not\_of\_rectangle\_segments}\ (S, SR)\);
9 \(CnR \leftarrow \text{not\_of\_rectangle\_components}\ (C, CR)\);
10 \(Obj \leftarrow R \cup SnR \cup CnR\);
11 \(LY \leftarrow \text{layout\_structure}\ (Obj)\);
12 \text{if judge\_doc\_of\_domain}\ (LY) = D \text{ then}\n   \(\triangleright\text{D is the domain of interest}\)
13 \text{if class\_recognition}\ (LY) = DM^p \text{ then}\n   \(\triangleright\text{Class}^p \text{ is the } p^{th} \text{ class of documents of the domain } D\)
14 \(\overline{Obj} = \text{Document\_Model\_matching}\ (DM^p, Obj)\);
   \(\triangleright\overline{Obj} \text{ is the list of the labelled objects of the current instance}\)
15 \text{else}\n16 \(\overline{Obj} = \text{model\_driven\_labelling}\ (Obj)\)
   \(\triangleright\text{Obj is the list of the unlabelled objects of the instance of the new class, }\overline{Obj} \text{ is the list of the}
   \text{labelled ones}\)
17 \((\text{DOC\_CLASS}^{\text{new}}, OBJ_0^{\text{new}}, \ldots, OBJ_i^{\text{new}}, \ldots) = \text{Add\_Document\_Model}\ (\overline{Obj})\);
   \(\triangleright DM^{\text{new}} \text{ is the new Document Model and } OBJ_i^{\text{new}} \text{ is the } i^{th} \text{ concept which represent the } i^{th}\n   \text{labelled object of the document of the new class}\)
18 \text{fi}\n19 \text{else}\n20 \text{document\_rejection}\ (image)\;
21 \text{fi}\n22 \text{while } \overline{R} \neq \emptyset \text{ do}\n   \(\triangleright\overline{R} \text{ is the list of labelled item rectangles in the set of labelled objects } \overline{Obj}\)
As discussed, the Conceptual Model can therefore be used both to process documents of known class, through the selection of the related Document Model, both to construct a new Document Model for an instance of an unknown class. Moreover, since the instances of each concept OBJ\(_p^i\) for the documents of the \(p^{th}\) class are attached to the slot “instances” of the related concept, and each concept OBJ\(_p^i\) is linked to the concept Cpt\(_j\) at the “named object level” which corresponds to its label, each instance of object is indirectly linked to the correspondent concept at the “named object level” too. This property can be used, for example, to submit a query to a database of documents. A query related to a concept at the “named object level” can be referred to the related object at the “geometry level” and to its instances.

4.3.3 The extraction of primitives: connected components

After having acquired a document instance by a scanner and converting it into an electronic format, the procedures to extract primitives have to be performed. Primitives are represented by connected components and, on their basis, by segments.

The function “connected_components (image)” is the first one to be performed by the procedure “document_processing” (§ 4.3.2). From the Conceptual Model point of view it means instantiating the concept CONN_COMP at the “primitive level” of abstraction.

According to the definitions in [16] in order to define a connected component some preliminary definitions have to be given.

**Definition 1** The eight nearest neighbor (8NN) pixels to a pixel \(P\) are the set of adjacent pixels to the pixel \(P\).

**Definition 2** A 8-path is a sequence of pixels \((P_0, P_1, ..., P_{n-1})\) so that for \(j > 1\), \(P_{j-1}\) is an 8NN pixel of \(P_i\), and for \(j < n - 1\) \(P_{j+1}\) is an 8NN pixel of \(P_i\).

A 8-path is a sequence of pixels \((P_0, P_1, ..., P_{n-1})\) so that the pixel at \([i_k, j_k]\) is a 8NN of the pixel at \([i_{k+1}, j_{k+1}]\) \(\forall 0 \leq k \leq n - 2\).

**Definition 3** A set of pixels is connected if for every pair of pixels \((P_i, P_j)\) of the set, there exists a path-8 with \(P_i\) as the first element and \(P_j\) as the final one.
Definition 4 A **connected component** is a set of black pixels connected with an 8-path.

In order to realize the extraction of the connected components from a black and white image a recursive procedure, which is intended to follow the 8-connected black pixels of a single one, can be realized.

Such a procedure scans row by row a binary image. When a black pixel is found it is assigned to a new component and the recursive procedure in Fig. 4.7 is performed.

```plaintext
follow (x,y)  
Begin  
P(x,y) = WHITE;  
if P(x+1, y) is BLACK then follow(x+1,y);  
if P(x+1, y+1) is BLACK then follow(x+1,y+1);  
if P(x+1, y+1) is BLACK then follow(x+1,y+1);  
if P(x-1, y+1) is BLACK then follow(x-1,y+1);  
if P(x-1, y) is BLACK then follow(x-1,y);  
if P(x-1, y-1) is BLACK then follow(x-1,y-1);  
if P(x, y-1) is BLACK then follow(x,y-1);  
if P(x+1, y-1) is BLACK then follow(x+1,y-1);  
end
```

Figure 4.7: **The recursive procedure to follow an 8-path.**

Such a procedure allows to follow a path in the direction of the first black pixel selected at each recursion. When a pixel is not adjacent to any more black pixels, the whole procedure returns back to the former pixel and a new direction is followed, and so on recursively, till the entire component is completely described in terms of the minimum and the maximum abscissa and ordinate.

This method is a particular form of clustering of objects, where the objects are represented by pixels and the condition of clustering is represented by the 8NN adjacency condition.

4.3.4 **The extraction of primitives: segment extraction and organization**

A feature of a document domain can be represented by the presence of horizontal and vertical segments to form rectangles which limit some item rectangles.

The extraction of segments is performed by the “segment_extraction” procedure on the basis of connected components.
Horizontal and vertical segments are extracted with techniques of low level processing [30]. Ordinates of horizontal segments can be organized, without repetitions, in a linked list whose elements contain such ordinates in an increasing order. Each element is associated with another linked list, whose elements contain starting and final abscissas of each horizontal segment related to the considered ordinate. The succession of two consecutive elements in the first level linked list represents the interval between the two sets of horizontal segments related to the ordinates of such elements. For this reason we define this structure as Interval List. The area bounded by the two sets of horizontal segments related to two consecutive ordinates is called vertical interval (Fig. 4.8).

Vertical segments are organized in a binary tree. Such a tree is called Interval Tree [31]; its leaves contain, without repetitions, abscissas of vertical segments. The succession of two consecutive nodes at the same level represents an interval (horizontal interval) between the two sets of vertical segments related to the abscissas of such nodes.

In [31] Interval Trees are used for rectangle description, and particularly to verify rectangle intersections over a linear dimension. In our work Interval Tree structure has been modified in order to consider all vertical segments which limit each horizontal interval, and to permit a linear parsing of leaves. Each leaf is linked to the next one and is associated with a linked list, whose elements contain starting and final ordinates of each vertical segment related to the current abscissa. An example of the structures we use is presented in Fig. 4.8.

The use of a more complex data structure for vertical segment collection is justified by the nature of the algorithm used for locating rectangles, described in § 4.3.5. Such an algorithm searches for vertical segments in a partition of the vertical interval, limited by the minimum and the maximum abscissa of the intersection of the projections on X-axis of the considered horizontal segments.

4.3.5 The algorithm for the extraction of item rectangles surrounded by segments

After a deskewing procedure, based on connected components and segments, the extraction of rectangles surrounded by segments (SEGM_ITEM_RC) can be executed (§ 4.3.2).

In this paragraph the algorithm for the extraction of rectangles surrounded by segment is described (Algorithm 4.2); Fig. 4.9 can be used for reference. A physical rectangle can be identified by four coordinates ($x_{min}$, $x_{max}$, $y_{min}$, $y_{max}$). From invoice domain point of view, a rectangle composed by segments is a physical area limited, at least, by two portions of horizontal segments. The two portions are to belong to different horizontal segments with different ordinates. The two portions are to have the same starting and final x-coordinate, respectively $x_{p\_start}$ and $x_{p\_stop}$, and their lengths have to exceed a threshold. Abscissas of such portions are obtained by the intersection, on X-axis, between
the projections of the current horizontal segment, with ordinate $y_i$, and the following ones with ordinate $y_j, y_j > y_i$. We denote by $H_{i,j}$ the $j^{th}$ horizontal segment of the $i^{th}$ ordinate.

The algorithm which performs segment item rectangle extraction is based on the individuation of such an intersection, that determines the coordinates $x_{p\text{-start}}$ and $x_{p\text{-stop}}$ of a partition of the current horizontal segment ($H_{i,j}$). In this rectangular area, identified by the coordinates $(x_{p\text{-start}}, x_{p\text{-stop}}, y_i, y_j)$, eventual further subdivisions into rectangles are individuated by the presence of vertical segments which completely cover the current vertical interval $(y_i, y_j)$. The abscissa $x_k$ of such a vertical segment determines an x-coordinate of a segment item rectangle.

The portion of the current horizontal segment individuated by the interval $(x_{p\text{-start}}, x_{p\text{-stop}})$ is allocated to the rectangles that it locates, and it is not considered any more.

In each vertical interval other eventual vertical segments, internal to a rectangle, which do not completely cover the current vertical interval are located.

All the extracted segment item rectangles are organized in a linked list. Each of these rectangles is represented by a node of the list; it contains x and y coordinates $(x_{min}, x_{max}, y_{min}, y_{max})$. The node of the list corresponding to a rectangle is associated with another linked list whose elements contain the coordinates of physical segments which limit such a rectangle and, in case, of the vertical internal segments. The algorithm for rectangle extraction is here below reported.

---

**Algorithm 4.2** \textsc{rectangles_surrounded_by_segments}($S$)
Figure 4.9: Intersection on X-axis of the projections of two horizontal segments

**Input**: List of segments ($S$);

**Output**: List of segment item rectangles ($SR$);

1. **Begin**
2. Horizontal segments are organized in an *Interval List*;
3. Vertical segments are organized in an *Interval Tree*;
4. Until all elements $y_i$ of the *Interval List* are visited;
5. **Begin**
6. Until all horizontal segments $H_{i,j}$ with $y_i$ ordinate are visited;
7. **Begin**
8. Until the current horizontal segment $H_{i,j}$ is completely allocated to a rectangle or all the segments $H_{u,v}$, $u > i$, $v > 0$ are visited;
9. **Begin**
10. The intersection of the projections on X-axis of $H_{u,v}$ and the portion of $H_{i,j}$ not allocated yet to a rectangle is calculated; such an intersection determines a partition of the current vertical interval; $y_i$ and $y_j$ are $y_{min}$ and $y_{max}$ of all rectangles in the current vertical interval;
4.3. THE PRAGMATICS OF THE NETWORK

11 In the current partition of vertical interval, all vertical segments are extracted
(abscessas of vertical segments which completely cover the current vertical inter-
val determine the $x_{\text{min}}$ and $x_{\text{max}}$ boundary abscissas of rectangles);

12 For each extracted rectangle, a new node of a list ($SR$), containing rectangle
coordinates, is constructed; the internal and boundary segments of the rectangle
are inserted in the list associated to the node list.

13 End
14 End
15 End
16 End

4.3.6 The algorithm for the extraction of item rectangles formed by
groups of connected components

After the individuation of rectangles surrounded by segments, from the areas outside
such rectangles, the item rectangles which are formed by a group of connected compo-
nents are selected. Item rectangles as group of components, represented by the concept
$\text{COMP}_{\text{ITEM}} \text{RC}$, derive from a procedure which is intended to group the extracted con-
nected components.

The algorithm of connected component clustering is based on the principle of the
connected component extraction method. It appears natural to extend such a procedure
to any other kind of objects that should be clustered. The connected component procedure
in fact represents, under this point of view, a particular case of the more general problem
of object clustering. In fact, in such a case, the clustering condition is represented by
the adjacency condition. Such a condition can just be satisfied only by the eight nearest
neighbors to the current pixel. Therefore the condition of clustering has to be estimated
only between the current pixel and each one of the eight nearest neighbors.

A generalization of this method has to consider that, in a set of objects, each one can
satisfy the condition of clustering with any other. Thus the “follow” procedure, described
in § 4.3.3 Fig. 4.7 is to be modified: the condition of clustering of a current object has to
be verified with respect to any other not assigned yet to any cluster.

Let us suppose that a set of objects is represented by a linked list. The “follow”
procedure for each element of the list will be modified. The modified function “new_follow”
is described here below (Algorithm 4.3), where $\text{first} \_\text{obj}$ is a pointer to the first object of
the list, $\text{pivot} \_\text{obj}$ is a pointer to the current object and $\text{last} \_\text{cluster}$ is a pointer to the
last cluster under consideration.

---

**Algorithm 4.3** NEW_FOLLOW (*first_obj*, *pivot_obj*, *last_cluster*)

**Input:** Link to the first object of the list of objects to be clustered, link to the current object, link to the last cluster;

**Output:** Link to the last cluster;

1. *pivot_obj* is marked as visited;
2. *obj* ← *first_obj*;
3. while (*obj* ≠ NULL) do
   4. if *obj* is not visited then
      5. if Condition between *pivot_obj* and *obj* is TRUE then
         6. *last_cluster* is updated;
         7. *last_cluster* ← new_follow (*first_obj*, *obj*, *last_cluster*);
      fi
   fi
   9. *obj* = *obj*.next;
11. Return *last_cluster*

---

The procedure “rectangles_as_groups_of_components” that invokes the “new_follow” procedure performs only a scanning of the list of the objects, each one indicated as “*pivot_obj*”. If such an object is not visited yet a new cluster is initialized and the current object is assigned to this one. Then the “new_follow” procedure is executed. The scheme of the “rectangles_as_groups_of_components” procedure is reported in the following algorithm (Algorithm 4.4).

---

**Algorithm 4.4** RECTANGLES_AS_GROUPS_OF_COMPONENTS (*C*)

**Input:** List of components (*C*);
4.3. THE PRAGMATICS OF THE NETWORK

**Output:** List of component item rectangles (CR);

1. \( \text{first.object} \leftarrow \mathcal{C}; \)
2. \( \text{pivot.object} \leftarrow \text{first.object}; \)
3. \( \text{first.cluster} \leftarrow \text{NULL}; \)
4. \( \text{last.cluster} \leftarrow \text{first.cluster}; \)
5. \( \textbf{while} \ (\text{pivot.object} \neq \text{NULL}) \ \textbf{do} \)
   6. \( \textbf{if} \ \text{pivot.object} \ \text{is not visited} \ \textbf{then} \)
      7. \( \text{a new cluster last.cluster is initialized;} \)
      8. \( \text{pivot.object is assigned to last.cluster;} \)
      9. \( \text{last.cluster} \leftarrow \text{new.follow (first.object, pivot.object, last.cluster);} \)
     10. \( \text{the list of clusters is updated;} \)
   11. \( \textbf{fi} \)
6. \( \textbf{od} \)
13. \( \text{pivot.object} \leftarrow \text{pivot.object.next;} \)
14. \( \text{CR} \leftarrow \text{first.cluster}; \)
15. \( \textbf{Return} \ \text{CR} \)

4.3.7 The Model Driven Labelling procedure

The function “model-driven labelling” is responsible of the instantiation of the concepts of the Conceptual Model at the “named object level” for a new class of documents or, equivalently, to create the new concepts \( \text{DOC\_CLASS}_\text{new} \) and \( \text{OBJ}_i^{\text{new}} \) at the “geometry level” for the new class.

From the Conceptual Model point of view, the Document Model construction is the problem of the instantiation of the concepts at the “named object level”.

In a multi-class domain of documents the problem is concerned with the instantiation of the physical objects extracted which are item rectangles, and segments and components which do not belong to item rectangles \((\mathcal{S}n\mathcal{R}, \mathcal{C}n\mathcal{R})\). Generally, the problem of instantiating \( \mathcal{S}n\mathcal{R}, \mathcal{C}n\mathcal{R} \) can be solved by selecting the instance which responds to the constraints required by the related concept. In more complicated cases, it can be solved with a procedure which is similar to the instantiation of item rectangles. The only difference is related to the marginal use of a keyword-based labelling procedure which is used only for item
rectangles. Therefore the following description refers to item rectangle labelling, but the same procedure can be easily extended to other extracted objects.

After all document item rectangles, have been extracted, the labelling procedure of them is performed.

The general procedure of locating each item rectangle is based on the following scheme:

1. A concept at the “named object level” is selected according to the Reading Plan;
2. the system searches for an item rectangle that represents the current concept suggested by the Reading Plan at the named object level;
3. the candidate item rectangles are selected. They represent the candidate instances of the concept, at the “geometry level” related to the concept at the “named object” one to be instantiated;
4. the judgement function of the current concept provides a credibility order among the selected item rectangles with respect to the physical and contextual constraints they satisfy;
5. the item rectangles selected are proposed to the user in a decreasing credibility order, till an item rectangle is accepted by the user for the current semantic label.

Credibility represents a rate of probability associated to an item rectangle to be an instance of the current concept to be instantiated. The item rectangle selected are grouped according to the physical and contextual constraints required by the current concept at the “named object level” to be instantiated. To such a concept \(Cpt\) a state array \(s(Cpt)\) can be associated. Each state of such an array represents a set of physical and contextual constraints an instance of the concept \(Cpt\) may assume: for example a range of values for the dimensions of the related instance, or the possible positions with respect to some objects considered as reference. The states of such array are obtained by the join of the states at the following variables:

- the Category of width or height which the object may belong to. They represent the possible clusters of a clustering procedure of item rectangles with respect to their width or height.

The probability values of their states are given by the slot “frequency” of the substructure “link description” of the link concrete of the concept at the “geometry level”, which is a possible concrete of the current concept at the “named object” one. The item rectangle instances of such concepts at the geometry level are obtained by a clustering procedure with respect to their width or their height. The related variables are \(WCat(Cpt)\) and \(HCat(Cpt)\) and their state arrays are \(wcat(Cpt)\), \(hcat(Cpt)\).
4.3. THE PRAGMATICS OF THE NETWORK

- the *Position* of the object with respect to some objects previously located or to particular coordinate values. The states of the position are given by the slot “Meanings” of the attribute “Position” related to the current concept. It can be represented by the variable $\text{Pos}$. The probability values of the qualitative positions are given by the array of the slot “Values” of the attribute “Position” of the related concept at the “named object level”. The state array can be indicated as $\text{pos}(\text{Cpt})$.

- the frequency of *Presence* of the object in a document. It is given by the slot “frequency” of the related frame, and it can be represented by the variable $\text{Pres}(\text{Cpt})$ and the state array is $\text{pres}(\text{Cpt})$.

In order to express all the possible positions of the concept with respect to other concepts previously labelled, any relational algebra can be used. For example Allen’s interval algebra [32] consisting of 13 relations $A = \{ \text{before (<), meets (m), overlaps (o), finished-by (fi), includes (di), starts (s), equals (=), started-by (si), during (d), finishes (f), overlapped-by (oi), meet-by (mi), after (>)} \}$ (Fig. 4.10), referred to horizontal and vertical directions and combined with respect to more than one object.

![Allen's 13 interval relations](image)

**Figure 4.10: Allen’s 13 interval relations**

The states of such variables consider all the possible constraints an object may satisfy.

The concept $\text{Cpt}_i$, whose frame description is reported in Fig. 4.11, is in a part-of relation with the the concept $\text{DOC_OF_DOMAIN}$. The slots “Concrete” declare the possible concepts which represent the representation of $\text{Cpt}_i$ at the “geometry” level. Four slots “Attribute” declare the coordinates of $\text{Cpt}_i$. The substructure “attribute description” declares, for each coordinate attribute, the type of the coordinate value (“Type_of_val”), the number of values (“Numb_of_val”), the coordinate restrictions (“Restriction”). Moreover,
such substructure has the slot “Comp_of_value”, which contains a link to a function that calculate the correspondent coordinate. The argument (“Arguments”) of such function is the coordinate of the correspondent concept \( OBJ_p \) of the current class \( p^{th} \). The current class \( p^{th} \) may be a known class (case of Document Model matching) or a new one (case of Document Model construction).

![Table](image)

Figure 4.11: The frame description for the concept \( Cpt_i \) at the “named object” level.

As discussed the state values of the variables \( HCat(Cpt) \), \( WCat(Cpt) \) and \( Pos(Cpt) \) are represented by the values of the arrays \( hcat(Cpt) \), \( wcat(Cpt) \), \( pos(Cpt) \). The states considered by the Conceptual Model for such variables represent all the possible physical and contextual constraints which an object may reveal itself with. Being \( n \), \( m \), \( p \) respectively the number of the states of the arrays \( hcat(Cpt) \), \( wcat(Cpt) \), \( pos(Cpt) \), the array of states \( s(Cpt) \) of the concept \( Cpt \) derives from the join of the states of the single arrays \( hcat(Cpt) \), \( wcat(Cpt) \) and \( pos(Cpt) \) (4.1).
4.3. THE PRAGMATICS OF THE NETWORK

\[ s(Cpt) = [hcat(Cpt) \times wcat(Cpt) \times pos(Cpt)] \]

\[ = [(hcat_0, wcat_0, pos_0), ... , (hcat_m-1, wcat_{p-1}, pos_{q-1})] \]

\[ = [s_0, s_1, ..., s_{n-1}], \quad n = m \cdot p \cdot q \]  

(4.1)

The probability of each state of \( s(Cpt) \) is given by the corresponding element of the matrix of the joint probabilities \( P(hcat \times wcat \times pos) \) (4.2).

\[ P(s(Cpt)) = P(hcat(Cpt) \times wcat(Cpt) \times pos(Cpt)) \]

\[ = [P(hcat_0) \times P(wcat_0) \times P(pos_0), ..., P(hcat_{m-1}) \times P(wcat_{p-1}) \times P(pos_{q-1})] \]

\[ = [P(s_0), P(s_1), ..., P(s_{n-1})], \quad n = m \cdot p \cdot q \]  

(4.2)

Furthermore, considering the state that corresponds to the event that the current concept is not present in a document of the current domain, the probability values of the state in \( s(Cpt) \) further depends on the variable \( Pres(Cpt) \).

It may assume the states in (4.3):

\[ s^*(Cpt) = [hcat \times wcat \times pos, N] = [s_0, s_1, ..., s_{n-1}, N] \]  

(4.3)

where the state “N” corresponds to the event “the concept \( Cpt \) is not present in a document of the current domain”, while the others correspond to the possible states the instance of the concept \( Cpt \) may assume when it is present in a document of the domain.

The probabilities of the states \( s^*(Cpt) \), conditioned to the probability of the states of the variable \( Presence (Pres) \) of the current concept in a document of the domain, with states in \( \underline{Pres} = [Y, N] \), are given by (4.4), (4.5):

\[ P(s^*(Cpt)/\underline{Pres}_0(Cpt) = Y) = [P(s(Cpt)), 0] \]  

(4.4)

\[ P(s^*(Cpt)/\underline{Pres}_1(Cpt) = N) = [0, 1] \]  

(4.5)

The conditional probability \( P(s^*(Cpt)/\underline{Pres}(Cpt)) \) is given by (4.6):

\[ P(s^*(Cpt)/\underline{Pres}(Cpt)) = \begin{bmatrix} \underline{Pres}_0 = Y & \underline{Pres}_1 = N \\ P(Cpt), 0 \end{bmatrix} [0, 1] \]  

(4.6)

The probability values of the variable \( Pres(Cpt) \) of states in \( \underline{Pres}(Cpt) = [P(Y), P(N)] \) are provided by the slot “Frequency” of the related concept.

The joint probability \( P(s^*(Cpt)), \underline{Pres}(Cpt)) \) is given by (4.7):
CHAPTER 4. A CONCEPTUAL MODEL BASED ON A SEMANTIC NETWORK

The result is given in (4.8):

\[
P(s^*(Cpt)) = \begin{bmatrix}
P(s_0) \cdot P(Y) & 0 & P(N) \\
P(s_1) \cdot P(Y) & 0 & P(N) \\
\vdots & \vdots & \vdots \\
P(s_{n-1}) \cdot P(Y) & 0 & P(N) \\
0 & 1 & P(N)
\end{bmatrix}
\]

Hence \(P(s^*(Cpt))\) is obtained by marginalizing \(P(s^*(Cpt), \text{pres}(Cpt))\) out of \(P(\text{pres}(Cpt))\).

The array \(P(s^*(Cpt))\) is calculated by the function “array_of_state_prob” (Algorithm 4.5), which performs the described computation of the joint probabilities for each concept at the “named object level”. Such a function is a procedure which is used by the procedure “model_driven_labelling” for each concept Cpt of the Reading Plan.

Algorithm 4.5 ARRAY_OF_STATE_PROB(hcat(Cpt), wcat(Cpt), pos(Cpt),
P(hcat(Cpt)), P(wcat(Cpt)), P(pos(Cpt)),
\text{pres}(Cpt), P(\text{pres}(Cpt)))

\textbf{Input:} Arrays of physical and positional relation states and arrays of their probabilities, array of presence states and array of their probabilities;

\textbf{Output:} Array of the state probabilities \(s^*(Cpt)\) associated to Cpt and the array of probabilities \(P(s^*(Cpt))\);

1. \(P(\text{hcat}(Cpt) \times \text{wcat}(Cpt) \times \text{pos}(Cpt)) = P(\text{hcat}_{i,j,k}) \cdot P(\text{wcat}_{i,j}) \cdot P(\text{pos}_{i,k}) \cdot P(\text{pos}_{q}) = P(\text{hcat}_0) \cdot P(\text{wcat}_0) \cdot P(\text{pos}_0), \ldots, P(\text{hcat}_{m-1}) \cdot P(\text{wcat}_{p-1}) \cdot P(\text{pos}_{q-1})])

2. \(P(s(Cpt)) = P(\text{hcat} \times \text{wcat} \times \text{pos})\), where \(P(s(Cpt)) = [P(s_0), \ldots, P(s_{n-1})], n = m \cdot p \cdot q;

3. Let \(s^*(Cpt) \leftarrow [s(Cpt), \text{pres}_1]

4. \(P(s^*(Cpt), \text{pres}(Cpt)) \leftarrow P(s^*(Cpt)/\text{pres}(Cpt)) \cdot P(\text{pres}(Cpt)) \leftarrow [P(s(Cpt), 0, (0, 1)] \cdot [P(Y), P(N)];\)
4.3. THE PRAGMATICS OF THE NETWORK

\[ P(s^*(Cpt)) \leftarrow \sum_{P(pres(Cpt))} P(s^*(Cpt), \overline{pres(Cpt)}) \]

\textbf{Return } s^*(Cpt), P(s^*(Cpt))

The probability value of each state \( s_{q(j)}^* \) is assigned to the item rectangles \( R_{s_{q(j)}^*} \) of the \( i^{th} \) \((i = q(j))\) list of item rectangles which are those assuming the correspondent state \( s_{q(j)}^* \) of the current variable \( Cpt \), related to particular physical and contextual constraints (Fig. 4.12). The value \( i = q(j), \ j = 0, ..., n - 1 \), is a permutation of \( i \) such that \( P(s_{q(j-1)}) \geq P(s_{q(j)}), \ j = 1, ..., n - 1 \).

\[
R_{s^*} = \begin{bmatrix}
R_{s_{q(0)}^*} & R_{s_{q(1)}^*} & \cdots & R_{s_{q(j)}^*} & \cdots & R_{s_{q(n-1)}^*}
\end{bmatrix}
\]

List of rectangles whose state is \( s_{q(0)}^* \)

List of rectangles whose state is \( s_{q(j)}^* \)

Figure 4.12: An array of the lists of item rectangles related to the states for a particular concept.

The item rectangles \( R_{s_{q(j)}^*}, \ j = 0, ..., n - 1 \) of each \( i^{th} \) list are proposed to the user in a decreasing credibility order, till an item rectangle is accepted for the current semantic label.

If an item rectangle is accepted for the current semantic label, that is the current concept at the “named object level” is instantiated, its constraints can be updated with the evidences of the related current concept instance (“concept_constraint_updating” procedure).

If none of the item rectangles selected is accepted by the user for the current semantic attribute, it means that the related object is not present in the current document class,
or no rectangle satisfies the constraints provided by the **Conceptual Model** for the current concept.

For the unlabelled left rectangles a procedure of keyword reading can be performed in order to assign them a semantic label ("keyword_based_labelling").

The scheme of the labelling procedure is in Fig. 4.13.

![Diagram](image)

**Figure 4.13: The scheme of the Model Driven Labelling procedure.**

The aim of the Model Driven Labelling procedure, however, is to resort to keyword locating and reading as less as possible (note that the keyword may not be present) and, in case keyword reading is necessary, it avoids consulting a large dimension dictionary for their validation. Moreover, it spares the user to indicate explicitly the physical objects to be labelled, and, in most cases, to write explicitly the semantic of the objects. Such an approach limits user interaction to only confirmation or refutation of a semantic attribution suggested by the system.

An example of object labelling is reported in § 5.5.3.

The general algorithm for the Model Driven Labelling procedure is here below reported (Algorithm 4.6).

**Algorithm 4.6** model_driven_labelling(\( \mathcal{R} \))

**Input:** the list \( \mathcal{R} \) of unlabelled item rectangles of the current instance of the new class docu-
4.3. THE PRAGMATICS OF THE NETWORK

\[ \text{Output: the list of labelled rectangles } \mathcal{R} \; ; \]

1. \( \mathcal{R}_{\text{first}} \leftarrow \mathcal{R} ; \)
2. \( \text{while } \mathcal{R} \neq \emptyset \text{ do} \)
3. \( \text{Cpt} \leftarrow \text{Selection of the first concept of the Reading Plan from the “named object level” of the Conceptual Model} ; \)
4. \( \text{Cpt}_{\text{first}} \leftarrow \text{Cpt} ; \)
5. \( \text{Let } hcat(Cpt) = [hcat_0(Cpt), ..., hcat_{m-1}(Cpt)] \leftarrow [\text{Cpt.Concrete.IT RC.hcat}(0), ..., \text{Cpt.Concrete.IT RC.hcat}(m - 1)] ; \)
6. \( \text{Let } P(hcat(Cpt)) = [P(hcat_0(Cpt)), ..., P(hcat_{m-1}(Cpt))] \leftarrow [\text{Cpt.Concrete.IT RC.hcat}(0).Frequency, ..., \text{Cpt.Concrete.IT RC.hcat}(m - 1).Frequency] ; \)
7. \( \text{Let } wcat(Cpt) = [wcat_0(Cpt), ..., wcat_{p-1}(Cpt)] \leftarrow [\text{Cpt.Concrete.IT RC.wcat}(0), ..., \text{Cpt.Concrete.IT RC.wcat}(p - 1)] ; \)
8. \( \text{Let } P(wcat(Cpt)) = [P(wcat_0(Cpt)), ..., P(wcat_{p-1}(Cpt))] \leftarrow [\text{Cpt.Concrete.IT RC.wcat}(0).Frequency, ..., \text{Cpt.Concrete.IT RC.wcat}(p - 1).Frequency] ; \)
9. \( \text{Let } pos(Cpt) = [pos_0(Cpt), ..., pos_{q-1}(Cpt)] \leftarrow [\text{Cpt.Position.Meanings}(0), ..., \text{Cpt.Position.Meanings}(q - 1)] ; \)
10. \( \text{Let } P(pos(Cpt)) = [P(pos_0(Cpt)), ..., P(pos_{q-1}(Cpt))] \leftarrow [\text{Cpt.Position.Values}(0), ..., \text{Cpt.Position.Values}(q - 1)] ; \)
11. \( \text{Let } pres(Cpt) = [pres_0(Cpt), pres_1(Cpt)] \leftarrow [Y, N] ; \)
12. \( \text{Let } P(pres(Cpt)) = [pres_0(Cpt), pres_1(Cpt)] \leftarrow [\text{Cpt.Frequency}, 1 - \text{Cpt.Frequency}] ; \)
13. \( \text{Let } g(Cpt) \leftarrow [hcat(Cpt) \times wcat(Cpt) \times pos(Cpt)] \leftarrow [(hcat_0, wcat_0, pos_0), ..., (hcat_{m-1}, wcat_{p-1}, pos_{q-1})] ; \)
14. \( \text{Let } g^*(Cpt) \leftarrow [g(Cpt), N(Cpt)] \leftarrow [s_0(Cpt), ..., s_{n-1}(Cpt), N(Cpt)], n \leftarrow m \cdot p \cdot q ; \)
where \( N(Cpt) \) represents the state “Cpt is not present in a document of the domain”;
15. \( \text{Let } P(g^*(Cpt)) \leftarrow [P(g(Cpt)), P(N(Cpt))] \leftarrow \text{array_of_state.prob}(hcat(Cpt), wcat(Cpt), pos(Cpt), pres(Cpt)) ; \)
16. \( \text{Let } \mathcal{R}_{g^*} \leftarrow \text{judge_Cpt.name_of_concept}(P(g^*(Cpt)), g^*(Cpt), \mathcal{R}) \) be the \( i^{\text{th}} \) list of can-
didate item rectangles where \( i = q(j), \) \( j = 0, ..., n - 1 \), is a permutation of \( i \) such that
\[
P(s_{q(j-1)}) \geq P(s_{q(j)}), \quad j = 1, ..., n - 1;
\]

for \( j = 0 \) to \( n - 1 \) do

\[\text{while } R_{s_{q(j)}} \neq \emptyset \text{ do}\]

\[\triangleright \text{ List of rectangles which satisfy a particular state } (s_{q(j)}^*) \text{ of the array } \tilde{s}^*(Cpt);\]

\[\text{Ok} \leftarrow \text{FALSE};\]

\[\text{Ok} \leftarrow \text{user_confirmation}(R_{s_{q(j)}^*}, Cpt);\]

\[\text{if Ok = TRUE then}\]

\[R_{s_{q(j)}^*}.\text{label} \leftarrow Cpt;\]

\[R_{s_{q(j)}^*}.\text{concrete_of} \leftarrow Cpt;\]

\[Cpt.\text{instance} \leftarrow R_{s_{q(j)}^*};\]

\[\text{break};\]

\[\text{fi};\]

\[R_{s_{q(j)}^*} \leftarrow R_{s_{q(j)}^*}.\text{next};\]

\[\text{od}\]

\[\text{fi};\]

\[\text{if Ok = TRUE then}\]

\[\text{concept_constraint_updating}(Cpt, s_i, P(g(Cpt)), 1);\]

\[\text{fi};\]

\[\text{od}\]

\[Cpt \leftarrow \text{Selection of a new concept according to the Reading Plan.}\]

\[\text{od}\]

\[R \leftarrow R_{\text{first}};\]

\[\text{keyword_based_labelling}(R);\]

\[\overline{R} \leftarrow R;\]

\[\triangleright \text{At this level all the item rectangles are labelled}\]

\[\text{Return } \overline{R}\]
4.3.8 Keyword based labelling

At the end of the Reading Plan, if some unlabelled item rectangles are left, a labelling procedure based on reading the keywords related to each rectangle, is executed (Algorithm 4.7). The reading phase is executed by an OCR system. Each OCR output is validated by a dictionary that contains the possible keywords belonging to the objects of the domain [33], [34].

This labelling phase, based on a bottom-up strategy, is partially driven by the Conceptual Model as well. In fact, when the Reading Plan is over, and there are some rectangles not labelled, we can have two possibilities:

1) all the concepts provided by the Reading Plan are assigned to an item rectangle. In this case the Conceptual Model doesn’t provide any label for the item rectangles left and it should be updated.

2) there are some concepts at the “named object level” not assigned to any rectangles. In this case the system attempts to match the keyword of the current unlabelled item rectangle with the alternatives suggested by the dictionary for the not assigned concepts. We can have two possibilities:

- the keyword matches with one of the dictionary alternatives related to a not assigned item rectangle: the related semantic label is assigned to the current item rectangle;

- it could be the case of a new keyword for a concept already present in the Conceptual Model, or the case of a keyword corresponding to a not present concept. Therefore the system shows the user all the not assigned concepts: if the user chooses one of them, only the dictionary is updated by the new keyword. If none of them is assigned to an item rectangle a new semantic has to be assigned, then the Conceptual Model and the dictionary have to be updated.

---

**Algorithm 4.7** KEYWORD_BASED_LABELLING ($\mathcal{R}$)

- **Input**: List $\mathcal{R}$ of all item rectangles (only the not labelled yet are considered);
- **Output**: List $\overline{\mathcal{R}}$ of completely labelled item rectangles;

1. while $\mathcal{R} \neq \emptyset$ do
2. 2.1 if $\mathcal{R}.label = \emptyset$ then
3. 2.2 Kw ← keyword_reading($\mathcal{R}$);
if Kw exists in a keyword dictionary then
  if Kw corresponds to Cpt.Name_of_concept then
    \(\text{keyword of an existing concept at the “named object level”}\)
    
    \(\text{Cpt} \leftarrow \text{selection of the concept related to the keyword;}\)
    \(\text{R.label} \leftarrow \text{Cpt.Name_of_concept;}\)
    \(\text{R.concrete.of} \leftarrow \text{Cpt}\)
    \(\text{Cpt.instance} \leftarrow \text{R};;\)
    
    \(s_i \leftarrow \text{constraint.extraction(R);}\)
    
    \(\text{such a function specifies the physical and contextual relations with respect to other}\)
    \(\text{concepts previously labelled}\)
    
    \(\text{concept.constraintupdating(Cpt, s_i, P(s(Cpt)), 1);}\)
  else
    \(\text{Keyword of a new Concept}\)
    
    \(\text{construction of a new frame } Cpt_{\text{new}} \text{ for the new concept;}\)
    
    \(\text{\{CM\} } \cup \text{\{Cpt_{new}\}}\)
    
    \(\text{Updating of the Conceptual Model;}\)
    
    \(\text{R.concrete.of} \leftarrow \text{Cpt}_{\text{new}};;\)
    \(\text{Cpt}_{\text{new}.\text{instance}} \leftarrow \text{R};\)
  fi
else
  \(\text{The keyword is not present in the dictionary}\)
  \(\text{R.label} \leftarrow \text{the user assigns the semantic label to the rectangle;}\)
  if R.label is the name of a concept not present yet in CM then
    \(\text{A new keyword of a new concept}\)
    
    \(\text{construction of a new frame } Cpt_{\text{new}} \text{ for the new concept;}\)
    
    \(\text{\{CM\} } \cup \text{\{Cpt_{new}\}}\)
    
    \(\text{Updating of the Conceptual Model;}\)
    
    \(\text{R.concrete.of} \leftarrow \text{Cpt}_{\text{new}};;\)
    \(\text{Cpt}_{\text{new}.\text{instance}} \leftarrow \text{R};\)
  else

4.3. The Pragmatics of the Network

A new keyword related to an existing concept

dictionary_updating (Kw);
R.concrete_of ← Cpt
Cpt.instance ← R;
si ← constraint_extraction (R);
concept_constraint_updating (Cpt, si, P(s(Cpt)), 1);
fi
fi

4.3.9 The concept constraint updating

The function “concept_constraint_updating” (Algorithm 4.8) is responsible of the updating of the constraints related to the concepts at the “named object level”. As discussed in § 4.3.7, each frame which describes a concept at the “named object level” provides the states of the possible physical constraints and of the possible contextual relations. The states which represent the physical constraints are obtained by the concepts at the “geometry level” attached to the slot “concrete”; the states of the possible contextual relations are obtained by the slot “Attribute”, which describes the variable Position. On their basis, the state array \( s(Cpt) \), associated to the current concept \( Cpt \) at the “named object level”, is obtained (§ 4.3.7).

If an object is labelled, it is identified the state of the array \( s(Cpt) \) which the object satisfies. Let \( s_i \) be the state of \( s(Cpt) \) the selected object satisfies. Since an element of such an array is obtained by the join of the states of the three arrays \( hcat(Cpt), wcat(Cpt) \) and \( pos(Cpt) \), the element \( s_i \) is itself a vector of three elements:

\[
s_i = (s_i(0), s_i(1), s_i(2)) = (hcat_j, wcat_k, pos_l).
\] (4.9)

Such elements are the states related to a category of height \( hcat_j \), of width \( wcat_k \) and a position \( pos_l \), of \( hcat(Cpt) \), \( wcat(Cpt) \) and \( pos(Cpt) \), respectively, which the object satisfies.

Since the probabilities of the states are estimated by the observed frequencies of the events, the new evidence can be used to update the probability values of the states of the arrays which \( s(Cpt) \) derives from.
Thus the updating of the probabilities of the states of each constraint array are obtained by the updating of the frequencies of the states, being $P(s_i) = P((hcat_j, wcat_k, pos_l)) = 1$ the new evidence for the current concept. Therefore the updating of the probability states is obtained as in 4.10 4.11 4.12.

$$P(hcat) = \frac{t \cdot [P(hcat_0), ..., P(hcat_j = s_i(0)), ..., P(hcat_{m-1})] + e_j}{t + 1}$$ (4.10)

$$P(wcat) = \frac{t \cdot [P(wcat_0), ..., P(wcat_k = s_i(1)), ..., P(wcat_{p-1})] + e_k}{t + 1}$$ (4.11)

$$P(pos) = \frac{t \cdot [P(pos_0), ..., P(pos_l = s_i(2)), ..., P(pos_{q-1})] + e_l}{t + 1}$$ (4.12)

where $e_j, e_k, e_l$ are the $j^{th}, k^{th}, l^{th}$ vectors of the canonical base in $\mathbb{R}^n, \mathbb{R}^p, \mathbb{R}^q$, and $t$ is the number of updating.

Afterwards the values of frequency of the slots “Concrete” and the probability values of the attribute “Position” are updated by the new values of frequency. Likewise the frequency of presence of the current concept (the slot $Cpt\_frequency$) is updated by the new evidence as in 4.13.

$$Cpt\_Frequency = \frac{t \cdot Cpt\_Frequency + 1}{t + 1}$$ (4.13)

---

**Algorithm 4.8** CONCEPT\_CONSTRAINT\_UPDATING ($Cpt, s_i, P(s(Cpt))$, $t$)

**Input:** the current concept $Cpt$ and the state $s_i$, the array $P(s(Cpt))$ of the probabilities of the states $s$, the number of updates;

1. $hcat_j \leftarrow s_i(0)$;
2. $wcat_k \leftarrow s_i(1)$;
3. $pos_l \leftarrow s_i(2)$;

\> $hcat_j$, $wcat_k$, $pos_l$ are the states of the variables HCat, WCat, Pos, respectively, satisfied by the labelled rectangle.

4. $P(hcat) \leftarrow \frac{t \cdot [P(hcat_0), ..., P(hcat_j = s_i(0)), ..., P(hcat_{m-1})] + e_j}{t + 1}$
5. $P(wcat) \leftarrow \frac{t \cdot [P(wcat_0), ..., P(wcat_k = s_i(1)), ..., P(wcat_{p-1})] + e_k}{t + 1}$
6. $P(pos) \leftarrow \frac{t \cdot [P(pos_0), ..., P(pos_l = s_i(2)), ..., P(pos_{q-1})] + e_l}{t + 1}$
4.3. THE PRAGMATICS OF THE NETWORK

\[ \triangleright e_j, e_k, e_l \text{ are the } j^{th}, k^{th}, l^{th} \text{ vectors of the canonical base in } \mathbb{R}^m, \mathbb{R}^p, \mathbb{R}^q \]

\[ \triangleright t = \text{number of updating} \]

7 for \( u = 0 \) to \( m-1 \) do

8 \( \text{Cpt.Concrete.IT_RC_hcat}(u).\text{Frequency} \leftarrow P(hcat_u) \)

9 for \( u = 0 \) to \( p-1 \) do

10 \( \text{Cpt.Concrete.IT_RC_wcat}(u).\text{Frequency} \leftarrow P(wcat_u) \)

11 for \( u = 0 \) to \( q-1 \) do

12 \( \text{Cpt.Position.Values}(u) \leftarrow P(pos_u) \)

13 \( \text{Cpt.Frequency} \leftarrow \frac{t \cdot \text{Cpt.Frequency} + 1}{t+1} \)
CHAPTER 4. A CONCEPTUAL MODEL BASED ON A SEMANTIC NETWORK

Figure 4.14: The generic layout of the semantic network for a multi-class domain.
Chapter 5

A case study: The design of an invoice processing system

5.1 Introduction

In this chapter an application for a particular multi-class document domain of the framework presented in § 3 and § 4 is described. The case study refers to the invoice domain. Such a domain has been chosen since it holds a relevant role in daily life, especially in commercial field. Many firms every day have to handle hundreds of invoices related to hundreds of commercial transactions, with a great waste of time and work. Therefore a system for automatic invoice processing is desirable. Moreover the invoice domain represents a typical example of a multi-class domain of documents, which the framework presented in § 3 and § 4 can be successfully applied to.

5.2 The invoice domain features

Invoices represent a universe of documents that may be divided into classes, with respect to the firm that issues the invoice. Even if invoices don’t have a fixed layout, the invoices of a firm are generally characterized by the invariance of logical and physical structure. A class associated to a firm is, therefore, a physical class in the sense of § 2.2. Moreover, a widely satisfied characteristic of invoices is represented by the presence of a logo which identifies the firm that issued the invoice. Therefore, a one-to-one correspondence between a physical class and a logo can be assumed.

The invoices belonging to a certain class consist of parts, which we call “objects”; each of them has a logical label and contains a particular type of data. In most cases objects are surrounded by segments to form open or closed rectangles. The objects we model in an invoice are: a logo which identifies the class of an invoice; upper, middle and lower section
which contain, respectively, data related to clients, products and totals; items that are rectangular regions which contain data and sometimes the corresponding keywords (see Fig. 5.1).

Classes of invoices present some structural or logical similarities, which can be captured by a general knowledge, that describes logical relationships among document objects and their physical constraints.

Nevertheless, these similarities, by themselves, are not sufficient to guarantee a robust model for understanding all the invoices. In fact, invoices have not the property of invariance of the qualitative positional relationship (“upper”, “lower”, “left of”, “right of”, and so on) among all their objects. Such a property would allow a document processing system to use successfully a model, based on positional relationships among objects [4] during the understanding phase, but in this case it can not be exploited. A general knowledge on domains characterized by positional relationships variance among objects is used in [5] during the understanding phase for probabilistic semantic attributions to the objects. Such an approach could be used in the invoice domain, but it would lead to a probabilistic object labelling. On the contrary, we want to design a system which has a deterministic knowledge on the position of an object.

In order to design a robust system to understand the instances of each invoice class and to guarantee a high degree of flexibility, it seems natural using the Semi-open architecture described in § 3 and § 4 which is characterized by a knowledge–based approach to document processing. For the invoice domain it will be based on specific invoice class models used for the understanding phase and a general knowledge, which describes the invoice universe, used as a framework for the phases of invoice analysis, classification, understanding and for the phase of Document Model construction.

5.3 A Semi-open architecture for an invoice processing system

As we have discussed, a class of invoices is characterized by a fixed logical and physical structure. This structure is represented by a specific model, the Document Model (§ 3.2.1), which describes all the objects appearing in the invoices of the class.

We have chosen a Document Model with relative coordinates because it allows the system to be tolerant to skew variation in the understanding phase.

An example of invoice and related Document Model is reported in Fig. 5.1.

A Document Model for each class of invoices can be used for the understanding phase of invoices of a known class. A Conceptual Model for the invoice domain can be also defined in order to provide a framework for invoice processing.

The Conceptual Model we define for the invoice domain is a declarative knowledge base, described by frames of the semantic network introduced in § 4. The Conceptual Model for
Figure 5.1: Example of invoice and related Document Model. In such a model the object “description” and the related coordinates \((x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}})\) are reported.

the invoice domain is described in § 5.4.

This kind of knowledge ensures flexibility to the system in the invoice domain. In fact, it can be used as a framework to process an invoice of a known class, by extracting the layout structure of an invoice (document analysis) and by associating (document classification) to the current invoice instance the Document Model of its class, in order to perform the document understanding phase. Otherwise the Conceptual Model is used as a base for the Document Model construction for an unknown class of invoices, in the spirit of capturing some features common to the various classes.

5.3.1 The invoice processing system

After acquiring an invoice instance by a scanner, and converting it into an electronic format, the system performs all the procedure of document analysis, intended to extract the layout structure of the current instance.

In particular connected components are extracted and, on their basis, segments and not segments components. Then a deskewing procedure, in order to align the coordinate system of the scanner to that of the current instance, is performed. Afterwards, the subdivision of rectangles which contain information, isolated segments and isolated components, can be accomplished. Therefore, the layout structure is extracted (§ 4.3.2).
Then, in order to classify the current instance (document classification), the system attempts to individuate and classify its logo [35], [36], [37], [38]. We have assumed the hypothesis, widely satisfied, of univocal mapping between logo and invoice class. As a consequence, the successful classification of a logo instance means having recognized the invoice class which the current invoice instance belongs to. On the contrary, if logo is not recognized, the system cannot map the invoice instance to its Document Model. In this case a new Document Model is to be built.

Depending on the successful or unsuccessful logo classification, the system provides two different alternatives.

1. Understanding phase of the current invoice instance.

   In such a case, the invoice can be understood directly by using a proper Document Model. Under Document Model supervision, the system can locate each object by matching the coordinates of the extracted objects the the coordinated of the correspondent objects of the Document Model. The coordinates of each object are related to two particular segments, which divide the three sections a generic invoice consists of.
2. Construction of a Document Model for the unknown invoice class and subsequent document understanding, as described in 1).

In such a case, the layout structure is extracted and the above mentioned three sections are located. Then, a procedure in order to assign a semantic label to each rectangle is executed: for example, the physical rectangles that contain description or total have to be determined. The labelling procedure [39], § 4.3.7 is driven by the help of the Conceptual Model representing a general knowledge about invoice domain. The result is a Document Model for the class of invoices which the current instance belongs to. Such a model can now be used for instance understanding.

The logical scheme of the whole system is reported in Fig. 5.2.

![Figure 5.3: A frame description of the concept INVOICE.](image)

**5.4 A Conceptual Model for invoice domain**

Invoice domain is characterized by a wide variety of fixed physical and logical structure classes, each of them mapped to its logo. Nevertheless, it is possible to provide the system with a general knowledge, for the whole domain, a Conceptual Model, oriented to capture some similarities among classes of invoices. The Conceptual Model describes invoice domain by a semantic network introduced in § 4; in particular, the Conceptual Model described for the invoice domain, results from the analysis of about thirty different classes of invoices (Fig. 5.11 can be considered as reference).
5.4.1 Invoice modelling at “named object” abstraction level

At the “named object” abstraction level, the concept INVOICE is the root; it is described by the frame in Fig. 5.3.

Each INVOICE can be divided into three sections: the middle section, represented by the concept MID_SECT, where products and their codes, prizes and related taxes are described; the upper section (UP_SECT), where data related to clients, their codes and conveyance modalities are reported; the lower section (LOW_SECT), where totals are summarized. Each of these sections is in a part-of relationship with INVOICE concept (see Fig. 5.11). A frame description for MID_SECT is presented in Fig. 5.4.

<table>
<thead>
<tr>
<th>Part of: INVOICE</th>
<th>Part: description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal_Node: DESCRIPTION</td>
<td>Frequency: 100%</td>
</tr>
</tbody>
</table>

| Part: total |
| Goal_Node: TOTAL | Frequency: 100% |

| Part: quantity |
| Goal_Node: QUANTITY | Frequency: 100% |

| Part: article_code |
| Goal_Node: ARTICLE_CODE | Frequency: 100% |

| Part: discount_prize |
| Goal_Node: DISCOUNT_PRIZE | Frequency: 80% |

| Part: list_prize |
| Goal_Node: LIST_PRIZE | Frequency: 100% |

| Part: tax |
| Goal_Node: TAX | Frequency: 75% |

| Part: unit_of_measure |
| Goal_Node: UNIT_OF_MEASURE | Frequency: 77% |

| Attribute: x_min |
| Type_of_val: INTEGER | Num_of_val: 1 |

| Attribute: y_min |
| Type_of_val: INTEGER | Num_of_val: 1 |

| Attribute: x_max |
| Comp_of_value: compute_coord |
| Arguments: SECT_RECT_M.x_min |

| Attribute: y_max |
| Comp_of_value: compute_coord |
| Arguments: SECT_RECT_M.y_min |

| Judgement: judge_middle_section |
| Arguments: SECT_RECT_M |

Figure 5.4: A frame description of the concept MID_SECT

Furthermore, each section is modelled as an aggregation of concepts which represent the specific items appearing in an invoice, as description, total, quantity and so on. An example of a frame of the concept TOTAL in a part-of relationship with MID_SECT is reported in Fig. 5.5. In such a frame the concepts, which are linked to TOTAL, are reported. The attribute “Position” contains the positional relationships (<h DESCRIPTION), ...,
>h(DESCRIPTION), written in terms of Allen’s algebra, which represent the possible position of TOTAL with respect to the concept DESCRIPTION in the horizontal direction.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type of val</th>
<th>Numb of val</th>
<th>Restriction</th>
<th>Comp of value</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_min</td>
<td>INTEGER</td>
<td>1</td>
<td>—</td>
<td>compute_coord</td>
<td>IT_RC_M_wcat(1),x_min</td>
</tr>
<tr>
<td>x_max</td>
<td>INTEGER</td>
<td>1</td>
<td>—</td>
<td>compute_coord</td>
<td>IT_RC_M_wcat(1),x_max</td>
</tr>
<tr>
<td>y_min</td>
<td>INTEGER</td>
<td>1</td>
<td>—</td>
<td>compute_coord</td>
<td>IT_RC_M_wcat(1),y_min</td>
</tr>
<tr>
<td>y_max</td>
<td>ARRAY OF FLOAT</td>
<td>13</td>
<td>[0,1]</td>
<td>TOTAL,constraint_updating</td>
<td>P(a*(TOTAL)), a*(TOTAL), DESCRIPTION, IT_RC_M_wcat(1)</td>
</tr>
</tbody>
</table>

Figure 5.5: The frame for the concept TOTAL.

### 5.4.2 Invoice modelling at “image”, “primitive” and “geometry level”

At “image”, “primitive” and “geometry level” the description of Conceptual Model for the invoice domain, coincides to the description of the correspondent levels of the Conceptual Model for a generic multi-class domain of documents § 4.2.

At the “image level” an invoice is represented by the concept IMAGE, whose parts are groups of connected pixels (CONN_PIXELS).

At the “primitive level” each group of connected pixels represents a connected component (CONN_COMP), which specialize in segments (SEGMENTS) and in not segment components (N_SEGM_COMP).

In their turn, not segment components specialize in components belonging (COMP_OF_RC) or not belonging (N_OF_RC_COMP) to rectangles which contains information, called item rectangles.
As described in § 4.2 components belonging to item rectangles are part of item rectangles without segments (RECT\_WHOUT\_SEGM), which are obtained by a clustering of connected components. Segments specialize in segments which belong or do not belong to item rectangles (N\_OF\_RC\_SEGM, SEG\_OF\_RECT). Segments belonging to item rectangles are part of RECT\_WT\_SEGM.

Segments which do not belong to an item rectangle are linked with a concrete of link to the concept \textit{ISOL\_SEGM} at the “geometry level” which represents a list of isolated components. As well, item rectangles with segments are linked to the concept \textit{SEG\_ITEM\_RC}, item rectangles without segments to \textit{COMP\_ITEM\_RC}, and components which do not belong to item rectangles are linked to \textit{ISOL\_COMP}.

\textit{SEG\_ITEM\_RC} and \textit{COMP\_ITEM\_RC} are specialization of item rectangles represented by the concept \textit{ITEM\_RECT}.

Therefore at the “geometry level” a generic invoice can be modelled by a collection of item rectangles, of isolated segments and isolated components whose generic features are reported in the frame which describes the concepts \textit{ITEM\_RECT}, \textit{ISOL\_SEGM} and \textit{ISOL\_COMP}. Such objects are part of the layout structure represented by the concept \textit{LAY\_STRUCT}.

As discussed in § 4.2 the part of the Conceptual Model which starts from the “image level” and crosses the “primitive level”, until this point of the “geometry level”, is used as a procedural scheme for the document analysis phase.

The remaining part of the “geometry level” is used for document classification and understanding, or, combined with the knowledge at the “named object level”, for the procedure of construction of a new Document Model for a unknown class.

In the part used for document classification and document understanding, the layout structure (\textit{LAY\_STRUCT}) specializes in a number of concepts \textit{INV\_CLASS}_p, equal to the domain classes known so far; \( p \) is the order number of the class. Each \textit{INV\_CLASS}_p consists of a particular set of objects. They are represented by the concepts \textit{INV\_OBJ}^p_i, where \( i \) is the index of a particular object of the \( p^{th} \) class. Moreover the concepts \textit{INV\_OBJ}^p_i are linked to the correspondent concept at the “name object level” with a concrete of link.

In the part used for the Document Model construction, depending on geometric features and on position in the invoice, item rectangles can specialize in \textit{ITEM\_RECT\_U}, whose instances, as parts, form the upper section rectangle, represented by the concept \textit{SECT\_RECT\_U}, in \textit{ITEM\_RECT\_M}, parts of the middle section rectangle (\textit{SECT\_RECT\_M}), and in \textit{ITEM\_RECT\_L}, parts of the lower section rectangle (\textit{SECT\_RECT\_L}). The frame description of the concept \textit{ITEM\_RECT\_M} is reported in Fig. 5.6.

\textit{SECT\_RECT\_U, M} and \textit{L} represent the concrete respectively of \textit{UP\_SECT}, \textit{MID\_SECT}, and \textit{LOW\_SECT} described at the “named object” abstraction level (Fig. 5.11).

\textit{ITEM\_RECT\_U} specializes in \textit{LOGO\_BOX}, which models the area that contains the invoice logo, \( IT\_RC\_U\_hcat(0) \), \( IT\_RC\_U\_hcat(1) \), \( IT\_RC\_U\_wcat(0) \), \( IT\_RC\_U\_wcat(1) \) and
5.4. A CONCEPTUAL MODEL FOR INVOICE DOMAIN

<table>
<thead>
<tr>
<th>ITEM_RECT_M</th>
<th>Attribute: y_min</th>
<th>Attribute: y_max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specialization-of: ITEM_RECT</td>
<td>Type_of_val: INTEGER</td>
<td>Type_of_val: INTEGER</td>
</tr>
<tr>
<td>Part-of: SECT_RECT_M</td>
<td>Number_of_val: 1</td>
<td>Number_of_val: 1</td>
</tr>
<tr>
<td>Restrictions: [MID_SECT.x_min, MID_SECT.x_max]</td>
<td>Restriction: —</td>
<td></td>
</tr>
<tr>
<td>Arguments: ITEM_RECT</td>
<td>Arguments: ITEM_RECT</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.6: The frame for the concept ITEM_RECT_M

IT_RC_U.wcat(2), which represent the rectangles of two different groups or categories of height, and of three different categories of width of the upper section rectangle.

The LOGO_BOX at the “geometry level” represents the concrete of the concept LOGO at the “named object” abstraction level.

The concepts IT_RC_U.hcat(i) and IT_RC_U.wcat(j) are linked, by a concrete of link, to the concepts at the “named object level” that are part of UP_SECT. SECT_RECT_U is linked, by a concrete of link, to the concept UP_SECT at the “named object level”. Similarly, ITEM_RECT_M and ITEM_RECT_L are described. For example, the item rectangles belonging to the middle section can be clustered only in three categories of width, represented by the concepts IT_RC_M.wcat(0), wcat(1), wcat(2), since usually their height is equal to the height of the middle section.

In general, the clustering of item rectangles in two different categories of height and in three different categories of width makes the labelling phase easier, because it leads to a preliminary selection of the possible item rectangles which may represent the object the system is searching for. The frame that describes IT_RC_M.wcat(1) is illustrated in Fig. 5.7.

On the whole the semantic network that describes the Conceptual Model for the invoice domain is illustrated in Fig. 5.11.
5.5 The pragmatics of the Conceptual Model of the invoice domain

As discussed in § 4.3, a Conceptual Model that describes a multi-class document domain, in this case the invoice domain, can be based can be used to process a known or an unknown class of invoices, by the procedure “document processing” § 4.3.

Rising the network from the “image level”, through the “primitive level”, till the “geometry level”, the system analyses (document analysis) the current document instance. As generally discussed in § 4.3, by extracting connected components (§ 4.3.3) and item rectangles (§ 4.3.5, § 4.3.6), thus instantiating the related concepts, the system extracts the layout structure of the invoice.

At this step, the concepts SEGMENT_ITEM_RC and COMPONENT_ITEM_RC are instantiated. Then isolated segments and isolated components can be selected, therefore the concept LAYOUT_STRUCT can be instantiated. At this level the procedure of document classification can be performed. This phase, which can be supported by the OCR system, is intended to verify whether the instance is an invoice and whether it belongs to a known class.

Since it is supposed a one-to-one mapping between a logo and an invoice class, if logo is recognized [36], [37], [40], the phase of document understanding can be executed, by matching the current instance of invoice with the Document Model of the related known class, as it is described in § 3.2.2.

On the contrary if logo is not recognized the system provides a procedure of semi-
automatic Document Model construction ("model_driven_labelling").

5.5.1 The construction of a new Document Model: the Model Driven Labelling procedure for invoices

As discussed in § 5.3.1, if logo is not recognized, the system provides a procedure of semi-automatic Document Model construction.

On the basis of the extracted segments and item rectangles, the three sections and the items are located.

The judgement functions used in the frames derive from the analysis of a sample set of about thirty different classes of invoices.

As discussed in and in § 4.3.7 the construction of a new Document Model results from the interaction between the knowledge described in the Conceptual Model at the "named object level", and the knowledge described at the "geometry level" of abstraction. Finally, user interaction is also required.

An important role is assumed by the presence of a Reading Plan (§ 4.3.1) at the "named object level". It is responsible of the selection of a concept at the such level of abstraction to be instantiated. In the Conceptual Model for invoice domain, the Reading Plan firstly selects the concept MID_SECT and then the other two concept representing the two sections left. Then the concepts which are part of each single section, are selected in a fixed order.

In § 5.5.2 and in § 5.5.3 the location of the three sections and the labelling of the rectangles appearing in middle section respectively, are discussed.

5.5.2 Location of the sections

The middle section is the first section to be located. Examining the frame that describes the related concept (MID_SECT, Fig. 5.4), as well as the Conceptual Model scheme in Fig. 5.11, it can be noted that such a concept has its concrete in SECT_RECT_M at the geometry level. Its judgement function, "judge_middle_section" whose argument is the instance of the concept SECT_RECT_M at the geometry level, simply returns such an instance with the label MID_SECT related to the "named object level". Therefore, it is sufficient to instantiate the concept SECT_RECT_M, at the "geometry level", to have the instance of the concept MID_SECT at the "named object" one.

In order to instantiate the concept SECT_RECT_M, the related frame provides the judgement function "judge_sect_rect_M".

A common feature of invoices is represented by the presence of two horizontal segments which separate the three sections. Therefore, simply the Interval List collecting all horizontal segments is the argument of "judge_sect_rect_M" function. The two horizontal segments, among the N of the collection, are selected by a heuristic criterion,
which provides of searching for two consecutive horizontal segments $H_1$ and $H_{t+1}$ of equal length (minus an error ($\epsilon$)), greater or equal than 80% of document width. Moreover the distance between $H_1$ and $H_{t+1}$ are to be the greatest distance between any other two consecutive segments, and such distance is to be greater or equal than 30% of the distance between the first and the last horizontal segment of the collection. In order to formalize these conditions, let $L_k = |x_{\text{start}}(H_k) - x_{\text{stop}}(H_k) + 1|$, be the length of segment $H_k$, and $d_{i,j} = |y(H_i) - y(H_j)|$, be the vertical distance between $H_i$ and $H_j$.

Then the following equations (5.1) to (5.4) summarize the foregoing conditions.

$$L_t = L_{t+1} \pm \epsilon; \quad \text{(5.1)}$$

$$L_t \geq 0.8 \ast W, \quad L_{t+1} \geq 0.8 \ast W; \quad W = \text{document width}; \quad \text{(5.2)}$$

$$d_{t,t+1} \geq d_{k,k+1}, \quad k = 0, 1, ..., N - 1; \quad k \neq t; \quad \text{(5.3)}$$

$$d_{t,t+1} \geq 0.3 \ast d_{0,N-1}. \quad \text{(5.4)}$$

The selected vertical interval $(y(H_t), y(H_{t+1}))$ is the MIDDLE SECTION.

In some classes of invoices a sequence of rectangles which contain the keywords of related rectangles of the middle section may be present. Each of these rectangles is in a vertical dependence relationship [10] with a rectangle of middle section, and all these rectangles are to have the same height (Fig. 5.8). Moreover all these rectangles are to be located in the same vertical interval.

In order to consider this possibility, the system verifies the presence of vertical dependences for the rectangles of the middle section with respect to the superior adjacent rectangles. If the condition is verified, rectangles of the superior sequence may be used to search for the keywords during the keyword-based procedure (§ refkeywords) of labelling for rectangles of middle section.

The segments which locate the three sections are elected as reference primitives, to be used during the understanding phase, having considered a Document Model with relative coordinates with respect to one of these two segments (§ 5.3.1).

The selected vertical interval $(y(H_t), y(H_{t+1}))$ gives the rectangle instance of the concept SECT_RECT_M at the “geometry level”, and, consequently, the instance of the concept MID_SECT at the “named object” one.

Once middle section has been located, the other two sections, upper and lower section, can be consequently individuated. Upper section is described, at the “named object level”, by the concept UP_SECT, whose concrete at the “geometry” one is represented by
5.5. THE PRAGMATICS OF THE INVOICE CONCEPTUAL MODEL

<table>
<thead>
<tr>
<th>Article code</th>
<th>Description</th>
<th>Qty</th>
<th>Prize</th>
<th>Amount</th>
</tr>
</thead>
</table>

Figure 5.8: Vertical dependence relationships among rectangles

$SECT\_RECT\_U$. Similarly lower section is described, at the “named object level”, by the concept $LOW\_SECT$, whose concrete, at the “geometry level”, is $SECT\_RECT\_L$.

The judgement functions appearing in $UP\_SECT$ and $LOW\_SECT$ have the same behaviour of the $MID\_SECT$ one. They associate to $UP\_SECT$ and $LOW\_SECT$, respectively, the rectangle instances of $SECT\_RECT\_U$ and $SECT\_RECT\_L$.

Thus, it is sufficient to instantiate such concepts at the “geometry level” to obtain the two rectangles that represent the objects upper and lower section. In order to instantiate the concept $SECT\_RECT\_U$, the related frame provides the judgement function “judge $sect\_rect\_U$” which determines the portion of image located over $H_f$ and over the rectangles of keywords, if exists. Such a rectangle is the instance of $SECT\_RECT\_U$ at the geometry level, and the instance of $UP\_SECT$ at the named object one. Likewise, the rectangle instance of $SECT\_RECT\_L$ and, consequently, the one of $LOW\_SECT$ is determined as the portion of image under $H_{f+1}$.

5.5.3 Location of the items

As discussed in 5.5.1, the concepts at the “named object level” can be ordered with respect to their contextual dependencies, to a decreasing observed frequency, and to the priority of reading. Such an order is the Reading Plan and gives a contribute for locating invoice objects in the labelling phase.

For each concept at the “named object level”, provided by the Reading Plan, the system searches for a rectangle which represents its concrete at the “geometry level”. Moreover such a rectangle has to satisfy the physical and positional constraints, evaluated by the judgement function of the concept at the “named object level” to be instantiated. Such a function gives a measure of certainty that the current extracted rectangle be an instance of the current concept.

Here below an example of the application of the “model-driven labelling” procedure related to the items of middle section is discussed. After the end of this procedure, some rectangles might be not labelled. In this case recognition of keywords must be performed for their labelling [39], § 4.3.8.
After the middle section has been located (§ 5.5.2), a procedure for clustering its rectangles is performed. Such a procedure is provided with a constraint related to the presence of only one rectangle related to the concept \textit{IT\_RC\_wcat(0)} of the “geometry level”.

According to the Reading Plan, description is the first item to be searched for. It is represented by the concept \textit{DESCRIPTION}, and its properties locate it univocally, without any reference to the positions of rectangles already labelled. Such an item results a minimally dependent concept [12], therefore it is the first item to be located in the labelling phase. According to \textit{DESCRIPTION} physical constraints, the system selects the rectangles related to the concept \textit{IT\_RC\_wcat(0)}. Since the clustering procedure provides only one rectangle in this cluster, such a rectangle is labelled as \textit{DESCRIPTION}.

The localization of other items is based on items previously located. In our case \textit{TOTAL} refers to the position of \textit{DESCRIPTION}; \textit{QUANTITY} refers to \textit{TOTAL} and \textit{DESCRIPTION} positions; \textit{ARTICLE CODE} refers to \textit{DESCRIPTION}, and so on with succeeding concepts. The order of localization of the items is suggested by the Reading Plan.

In order to locate items, the system searches for a rectangle, related to the current concept suggested by the Reading Plan. The rectangles, instances of the concept at the “geometry level” that is the concrete of the concept at the “named object level” to be instantiated, are selected. Among them, a number of rectangles, satisfying some constraints of the current concept at the “named object level”, are selected. The number of constraints satisfied, evaluated by the current concept judgement function, provides a credibility order among the rectangles selected. The rectangles selected are proposed to the user in a decreasing credibility order, till a rectangle is accepted for the current semantic label.

For example the judgement function for the concept \textit{TOTAL} gives a measure of certainty that the extracted candidate rectangle be an instance of \textit{TOTAL}. Such a function considers the rectangles of middle section that satisfy to particular physical constraints (the related frame is described in § 5.4.1 Fig. 5.5), and their position with respect to the rectangle already labelled as an instance of \textit{DESCRIPTION}.

It can be observed that the item \textit{quantity} may satisfy different physical constraints, and has different adjacency relationships with other items depending on the class of the invoice; its concrete concepts are represented by \textit{IT\_RC\_wcat(1)} or \textit{IT\_RC\_wcat(2)} at the “geometry level”. The related probabilities of representing the concept \textit{QUANTITY} are given by the slot “Frequency” of the substructure that describes its concrete concepts. For simplicity only these two alternatives for the category of width are considered in the state variable \textit{wcat(Qty)} = [\textit{wcat}_1, \textit{wcat}_2]. Moreover the item \textit{quantity} may appear on the right or on the left of the item \textit{description}. These two alternatives are given by the states of the attribute \textit{QUANTITY}.Position representing all the possible contextual relations the concept \textit{QUANTITY} may satisfy. As well, only these two different alternative
positions are reported in the slot “Meanings” which represents the states of the variable QUANTITY.Position, expressed in terms of Allen’s algebra for the horizontal direction (Fig. 5.9). The related probabilities are given by the slot “Values” of the attribute QUANTITY.Position (Fig. 5.9).

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>Attribute: x_max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part-of: MID_SECT</td>
<td>Type_of_val: INTEGER</td>
</tr>
<tr>
<td>Concrete: geometry</td>
<td>Numb_of_val: 1</td>
</tr>
<tr>
<td>Goal_node: IT_RC_wcat(1)</td>
<td>Restriction: —</td>
</tr>
<tr>
<td>Frequency: 70%</td>
<td>Comp_of_value: compute_coord</td>
</tr>
<tr>
<td>Goal_node: IT_RC_wcat(2)</td>
<td>Arguments: ITEM_RECT.x_max</td>
</tr>
<tr>
<td>Frequency: 30%</td>
<td></td>
</tr>
<tr>
<td>Attribute: x_min</td>
<td></td>
</tr>
<tr>
<td>Type_of_val: INTEGER</td>
<td></td>
</tr>
<tr>
<td>Numb_of_val: 1</td>
<td></td>
</tr>
<tr>
<td>Restriction: —</td>
<td></td>
</tr>
<tr>
<td>Comp_of_value: compute_coord</td>
<td></td>
</tr>
<tr>
<td>Arguments: ITEM_RECT.x_min</td>
<td></td>
</tr>
<tr>
<td>Attribute: y_min</td>
<td></td>
</tr>
<tr>
<td>Type_of_val: INTEGER</td>
<td></td>
</tr>
<tr>
<td>Numb_of_val: 1</td>
<td></td>
</tr>
<tr>
<td>Restriction: —</td>
<td></td>
</tr>
<tr>
<td>Comp_of_value: compute_coord</td>
<td></td>
</tr>
<tr>
<td>Arguments: ITEM_RECT.y_min</td>
<td></td>
</tr>
<tr>
<td>Attribute: y_max</td>
<td></td>
</tr>
<tr>
<td>Type_of_val: ARRAY OF FLOAT</td>
<td></td>
</tr>
<tr>
<td>Numb_of_val: 2</td>
<td></td>
</tr>
<tr>
<td>Restriction: [0,1]</td>
<td></td>
</tr>
<tr>
<td>Meanings: £_h(DESCRIPTION), &gt;_h(DESCRIPTION)</td>
<td></td>
</tr>
<tr>
<td>Values: {0.2, 0.8}</td>
<td></td>
</tr>
<tr>
<td>Comp_of_val: QUANTITY_constraint_updating</td>
<td></td>
</tr>
<tr>
<td>Arguments: concept_constraint_updating</td>
<td></td>
</tr>
<tr>
<td>Judgement: judge_quantity</td>
<td></td>
</tr>
<tr>
<td>Argument: P(s*(QUANTITY)), s*(QUANTITY)</td>
<td></td>
</tr>
<tr>
<td>IT_RC_wcat(1), IT_RC_wcat(2)</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION, TOTAL</td>
<td></td>
</tr>
<tr>
<td>Frequency: 100%</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.9: The frame of the concept QUANTITY.

The function “array_of_state_prob”, invoked for the concept QUANTITY, calculates the array \(P(s^*(Qty))\) related to such a concept.

Therefore, being

\[
P(wcat(Qty)) = [P(wcat_1), P(wcat_2)] = \\
= [Qty.Concrete.IT_RC.wcat(1).Frequency, Qty.Concrete.IT_RC.wcat(2).Frequency]
\]

\[
= [0.7, 0.3]
\]
$$P(pos(Qty)) = [P(pos_0), P(pos_1)] =$$
$$= [Qty.Position.Values(0), Qty.Position.Values(1)] =$$
$$= [0.2, 0.8]$$

the joint probability $P(pos, cat)$ results (5.7)

$$P(pos(Qty), cat(Qty)) =$$

\[
\begin{array}{cc}
pos_0 & wcat_1 \ 0.2 \times 0.7 & 0.2 \times 0.3 \\
pos_1 & wcat_2 \ 0.8 \times 0.7 & 0.8 \times 0.3 \\
\end{array}
\]

$$=\begin{bmatrix} wcat_1 & wcat_2 \ pos_0 & 0.14 & 0.06 \\
pos_1 & 0.56 & 0.24 \end{bmatrix}$$

(5.7)

Defining the new array of states $s(Qty)$ for the concept $Qty$, as well as the general array of state $s(Cpt)$ for the concept $Cpt$, that may assume the following states (5.8):

$$s(Qty) = [s_0, s_1, s_2, s_3] =$$

$$= [wcat_1pos_0, wcat_1pos_1, wcat_2pos_0, wcat_2pos_1]$$

(5.8)

the probabilities of the states of $s(Qty)$ are given by the corresponding elements of the matrix of the joint probabilities $P(pos(Qty), cat(Qty))$ (5.9)

$$P(s(Qty)) = [P(s_0), P(s_1), P(s_2), P(s_3)]$$

(5.9)

Furthermore, considering the state that corresponds to the event that the concept $QUANTITY$ is not present in an invoice (state “N”), a new array $s^*(Qty)$ which contains also the state “N” will have the following states (5.10):

$$s^*(Qty) = [s_0, s_1, s_2, s_3, N]$$

(5.10)

The probabilities of the states of the new array, conditioned to the probability of the states variable $Presence (pres)$ of the current concept in an invoice, are given by (5.11), (5.12):

$$P(s^*(Qty)/pres_0 = Y) = [P(s(Qty)), 0]$$

(5.11)

$$P(s^*(Qty)/pres_1 = N) = [0, 1]$$

(5.12)

The conditional probability $P(s^*(Qty)/pres)$ is given by (5.13):

$$P(s^*(Qty)/pres) = \begin{bmatrix} pres_0 = Y & pres_1 = N \\ [P(s(Qty)), 0] & [0, 1] \end{bmatrix}$$

(5.13)
The probability values of the state of Presence, \( P(\text{pres}) = [1, 0] \) correspond to the probability of presence of the concept \( QUANTITY \) in an invoice of 100%.

The joint probability \( P(\mathbf{z}^*(\text{Qty}), \text{pres}) \) is given by (5.14):

\[
P(\mathbf{z}^*(\text{Qty}), \text{pres}) =
\begin{array}{c|c}
\text{pres} & \text{pres} = Y & \text{pres} = N \\
\hline
s_0 = \text{wcat}_1 \text{pos}_0 & P(s_0) \ast 1 & 0 \ast 0 \\
s_1 = \text{wcat}_1 \text{pos}_1 & P(s_1) \ast 1 & 0 \ast 0 \\
s_2 = \text{wcat}_2 \text{pos}_0 & P(s_2) \ast 1 & 0 \ast 0 \\
s_3 = \text{wcat}_2 \text{pos}_1 & P(s_3) \ast 1 & 0 \ast 0 \\
s_4 = N & 0 \ast 1 & 1 \ast 0
\end{array}
\]

Hence \( P(\mathbf{z}^*(\text{Qty})) \) is obtained by marginalizing \( P(\mathbf{z}^*(\text{Qty}), \text{pres}) \) out of \( P(\text{pres}) \). The result is given in (5.15):

\[
P(\mathbf{z}^*(\text{Qty})) = [P(\mathbf{z}^*(\text{Qty})), 0] =
\begin{bmatrix}
P(s_0), P(s_1), P(s_2), P(s_3), 0
\end{bmatrix} =
\begin{bmatrix}
0.14, 0.56, 0.06, 0.24, 0
\end{bmatrix}
\]

Each extracted rectangle of middle section is provided with the value of probability of the state it assumes. Then the rectangles are presented to the user in a decreasing probability order (first of all the rectangles which assume the state \( s_1 \), that are those on the right of description and belonging to the \( \text{wcat}_1 \) category cluster of width, then those on the right of description and belonging to the \( \text{wcat}_2 \) category cluster of width, and so on) till a rectangle is assigned to the concept \( QUANTITY \).

User interaction, in confirmation or refutation of a labelling hypothesis, is justified by the fact that rectangle semantic labelling is a “key” procedure in a Document Model construction. An error in the labelling phase of a rectangle, in fact, spreads to the others, with the risk of making every other semantic attribution fail.

If none of the rectangles selected is accepted by the user for the current concept, it means that the related item is not present in the current invoice class, or it is present but no rectangle satisfies the constraints provided by the Conceptual Model for it. In both cases the Conceptual Model should be updated, in the first case, with respect to the frequency of the current concept, or, in the second one, with the new constraints for it.

Similarly, the new evidence of the object quantity are used to update the constraints of the concept \( QUANTITY \) as discussed in 4.3.9.

At the end of the Reading Plan, if some unlabelled rectangles are left, a labelling procedure, based on reading keywords, is going to start. This procedure is based on an
OCR system and it uses a proper keyword dictionary § 4.3.8.

The scheme of the labelling procedure is in Fig. 5.10.

Figure 5.10: Scheme of item rectangle labelling procedure
5.5. THE PRAGMATICS OF THE INVOICE

CONCEPTUAL MODEL

Figure 5.11: The semantic network that describes invoice Conceptual Model.
Chapter 6

A font adaptive modular OCR system

6.1 Introduction

In document processing research area, characters recognition is one of the critical problems to tackle.

The systems that can be found in literature usually rely upon commercial OCR (Optical Character Recognizer) systems. The disadvantage of such systems is that they are not well suited for particular classes of documents and for their specific fonts. This fact can sometimes determine a certain weakness and lack of reliability as far as the commercial OCR systems is concerned, making the reading task fail.

A neural-based OCR system for processing documents of a multi-class domain has been realized, in order to adapt the OCR to the domain of interest.

Recently, in such an area, particular methods based on the combination of some classifiers have been proposed in order to improve the rate of recognition of single units of classification.

According to [41], the output information that various classification algorithms supply or are able to supply can be divided into three levels.

1. The abstract level: a classifier only outputs a unique label \( j \), or for some extension, it outputs a subset of labels.

2. The rank level: a classifier ranks all the labels in a queue with the label at the top being the first choice.

3. The measurement level: a classifier attributes each label a measurement value to address the degree that the pattern has the label.
CHAPTER 6. A FONT ADAPTIVE MODULAR OCR SYSTEM

The third level contains the highest amount of information, the first one contains the lowest.

This work proposes the comparison of three kind of systems for pattern classification based on neural technique:
1) Classification with neural networks used as autoassociators;
2) Classification derived from a hierarchical combination of autoassociators and discriminating networks between two classes (*Championship Algorithm*);
3) Classification derived from a Multilayer Perceptron (MLP) classifier among all classes combined with the *Championship Algorithm*.

The classification methods proposed in this work provide a rank level of information, since they rank the various classes with no other information and the first one is selected as the right class.

All the results are related to a database of characters that has been constructed to the purpose. For each image in the database a variable number of patterns has been generated by overlapping a synthetic noise on the model of Baird [42]. The use of a synthetic noise to be overlapped to character images allows not to have a database of large dimensions and to generate patterns dynamically.

Moreover the problem of character segmentation has been faced. This problem is not present in experiments on characters of the database (off-line phase), but it becomes crucial when the OCR system works on new patterns, extracted from documents whose information has to be archived (on-line phase). Many different approaches have been proposed to solve such a problem: projection analysis, connected component procedure, bounding box analysis, dissection by intersection analysis. In the whole, segmentation methods can be listed under four main sections ([43]):

1. The first one consists of methods that partition the input image into sub-images, which are then classified. Such a method is called “dissection” [44], [24]. Interaction with classification is restricted to re-processing of unacceptable recognition results.

2. The second one avoids complex dissection algorithms, and segments the image either explicitly or implicitly: classification is used to select from segmentation candidates, generated in an exhaustive fashion (recognition-based segmentation). The explicit segmentation consists of those methods that segment the image blindly at many boundaries chosen without regard to image features and then try to choose an optimal segmentation by evaluating the classification of the sub-images generated ([45]). The implicit one comprises methods that represent the physical location of image features, and seek to segment this representation into well-classified subsets (methods based on Hidden Markov Models or on letter features representation).

3. The third one is a combination of the first two, employing dissection together with re-combination rules to define potential segments, but using classification to select
6.2 DATABASE CONSTRUCTION

from the range of admissible segmentation possibilities offered by such sub-images [14].

4. The four one consists of holistic approaches that avoid segmentation by recognizing entire character strings as units [34]. Such methods follow a two-step scheme:
   - the first step performs feature extraction;
   - the second step performs global recognition by comparing the representation of the unknown word with those of the references stored in a lexicon.

The first two methods are concerned with two radically different segmentation strategies. The first one attempts to choose the correct segmentation points by a general analysis of image features. The other carries out no dissection of the image, but simply do a form of model-matching against image contents. While the third is a combination of the first two, as discussed, the last one represents a completely different approach to word recognition: it considers the entire word as the object to recognize to by-pass the problem of dissection of characters. The recognition phase is usually performed by the help of a dictionary.

In this work for the phase of recognition an iterative segmentation procedure, based on connected component extraction with adaptive binarization threshold, is proposed.

This chapter is organized as follows: in § 6.2 the construction of database is illustrated. § 6.3 describes pattern generation by the overlapping of Baird’s synthetic noise model [42]. In § 6.4 the classification methods and their results are reported. § 6.5 illustrates the segmentation procedure during the understanding phase of a document. Finally, in § 7 some conclusions are reported.

6.2 Database construction

The preliminary phase of the realization of an OCR system which is intended to be adaptive as far as character font is concerned, is represented by the construction of a database of characters (digits, lowercase letters and uppercase letters), related to the particular font of the document domain of interest.

Such a database may also be divided with respect to different font of characters. The division with respect to font allows to use such a knowledge in order to train specified neural networks dedicated to recognize characters of a particular font.

The extraction of characters has been developed by an X-interface program which makes the labelling phase easier. The program that performs such an extraction consists of four steps:

1. Deletion of the not useful parts of the image;
2. Binarization of the image;
3. Connected components extraction;
4. Labelling of the selected components.

Figure 6.1: Acquisition phase for database creation

1. Deletion of the not useful part of the image: such a procedure only allows to speed up the procedure of connected component extraction. It reduces the number of connected components from the image, by keeping the only useful parts which contains the desired components. For example lines, logo, characters of different fonts from the desired ones are deleted from the image by a procedure which erase the content of a region selected by the mouse.

2. Image binarization: by this procedure a grey-level image is converted in a black and white one. Such a procedure is a preliminary step to the connected component extraction. The binarization procedure is provided with a binarization threshold. In
6.3. **PATTERN GENERATION**

In this work, being \textit{black} = 255 and \textit{white} = 0, it is used a low threshold \((Bin\_Thres \leq 128)\) in order to lose as less quantity of information as possible from each character.

3. Connected component extraction: this is a widely used procedure ([46], [47], [48]) that allows to locate primitive objects, generally characters, parts of characters and graphic elements. Since it has been chosen a low binary threshold, the connected component extraction procedure may select more than one character for each component. Generally, it is a problem during the \textit{on-line phase}, but, since this segmentation program allows the user to change some parameters, in this phase (\textit{off-line phase}) it is possible to separate touching characters by increasing the binary threshold and by performing a new connected component procedure in the selected windows, till a satisfactory component from a perceptive point of view is obtained.

4. Labelling of the selected component: once a satisfactory component is selected, it is classified by a label and saved in the related directory. Otherwise, if the current component does not contain a character but simply some noise, such a component is rejected.

Globally, the images of each class are organized in two sets, about \(2/3\) for the learning set and about \(1/3\) for the test set.

In Figs. 6.1, 6.2 are reported two phases of the database creation, executed by the help of an X-interface: the phase of image acquisition and the connected component classification after having erase the not useful parts of the image.

### 6.3 Pattern generation

Images of the character database are used to generate patterns for each class of characters, which neural networks are to be fed on, both for the phase of learning, both for the test one. Since a not standard database is used, but it can be dynamically constructed depending on the font of characters that have to be processed, it is possible to obtain a not homogeneous database as far as the number of instances among classes is concerned. Moreover, it is desirable that the number of patterns, which neural networks are fed on for the learning phase, be homogeneous for each class. An homogeneous training set, with respect to the number of patterns for each class, leads to a more reliable learning phase, avoiding a better learning of patterns of a class rather than of an another one. For this reason, for each character image in the database, a variable number of patterns has been generated by overlapping a synthetic noise as far as the model proposed by Baird [42] suggests.

Baird’s model is a descriptive one, therefore there does not always exist a one-to-one correspondence between a noise parameter and a cause of noise in images. Moreover it is
a local model which requires ideal black and white images with high resolution. In such an application we have grey level images that, under the same resolution, provide more information than an equivalent black and white one.

Given in input a grey level image and a set of parameters, a corrupted grey level image is generated by the following sequence of procedures.

1. A frame of 3 pixel is added to the original image, so that the following procedures do not corrupt image completeness.

2. The image is expanded with a two step interpolation procedure. In the first step image dimensions are three times increased, by copying the value of each pixel in a 3x3 window of the output image. In the second step the value interpolation is performed. To each pixel of the new image the average of the values of pixels in the 3x3 window, centered on such a pixel, is calculated. The result is an image where
6.3. PATTERN GENERATION

the center pixels of the 3x3 windows are equal of the original image ones, while other pixels have intermediate values.

3. The grey level image is rotated. This procedure simulates the noise produced by possible inclinations that may occur in the acquisition procedure. The rotation angle \(\text{skew}\) is calculated by a uniform probability distribution in the range defined by two parameter \(\alpha_{\text{min}}\) and \(\alpha_{\text{max}}\) \((\text{skew} \in [\alpha_{\text{min}}, \alpha_{\text{max}}])\). In our experiments we used a value \(\text{skew} \in [-1^\circ, +1^\circ]\).

4. The noise produced by possible translations is simulated. Translations of only one pixel are considered. The effect is particularly evident for edge pixels. Each edge pixel may be more or less white in dependence of the portion of the edge pixel contained in the sensor of the scanner related to such a pixel. The model does not provide any parameter, the image is shifted with casual values in horizontal and vertical direction of at most one pixel.

5. Optical distortions added by the scanner are modelled by an image blurring. In [42], the point spread function is modelled by a circularly symmetrical Gaussian filter. The Gaussian filter has a \(\text{blur}\) standard error measured in terms output pixels. Basically, the convolution between Gaussian function \(g(x, y)\):

\[
g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}
\]

where \(\sigma = \text{blur}\), and the image is calculated. The \(\text{blur}\) value is casually chosen in correspondence to the generation of every noisy image.

6. Image pixel values are expanded dynamically in the range of grey level values \(([0, 255])\).

7. The image is scaled to get the final resolution. The image is reported to the initial dimension, basically by the inverse procedure of expansion. Each new pixel is calculated by the average of the pixel values of the expanded image related to the new pixel area.

8. A procedure that simulates the variability of sensors sensibility and that execute a binarization procedure, is performed. This procedure is performed by increasing or decreasing the grey-level value of the pixels before the binarization step. The level of variation is determined by generating a value for each pixel which derives from a Gaussian function of the parameter \(\text{sens}\), \(\text{a priori}\) assigned, multiply per 255 and added to each pixel. Then the binarization by a random value of threshold for each image is performed.
In Fig. 6.3 the effects of the described noisy procedures with respect to particular values of parameters are shown.

Figure 6.3: An original character image and its modification by overlapping some components of synthetic noise.

Once a character image has been corrupted by overlapping the synthetic noise components, it is to be normalized within fixed dimensions. Such values determine the dimension of the patterns which feed the classification networks.

Let $C_h$, $C_w$ be respectively the height and the width of a component and $NC_h$, $NC_w$ respectively the height and the width of the normalized one, so that $NC_w \times NC_h$ is the dimension of the pattern which feeds a neural network. Firstly each image is expanded, if it is necessary, in order to have:
6.3. PATTERN GENERATION

\[ C_h > N C_h \quad C_w > N C_w \]  \hspace{1em} (6.2)

Let

\[ R = \frac{C_w}{C_h} \quad R_f = \frac{N C_w}{N C_h} \]  \hspace{1em} (6.3)

respectively be the ratio width/height of the component and the ratio width/height of the final image from which the pattern is obtained (normalized component). In order not to introduce any distortions, such ratios should be the same. Generally this condition is not verified. For this reason a new value of the dimension to be increased is calculated, as it is illustrated in Eqs. (6.4), (6.5).

If \( R > R_f \)

\[ C_{h_{\text{new}}} = \frac{1}{R_f} * C_w \]
\[ C_{w_{\text{new}}} = C_w \]  \hspace{1em} (6.4)

else

if \( R < R_f \)

\[ C_{w_{\text{new}}} = R_f * C_h \]
\[ C_{h_{\text{new}}} = C_h \]  \hspace{1em} (6.5)

Since the new value of a dimension (\( C_{h_{\text{new}}} \) or \( C_{w_{\text{new}}} \)) generally is not an integer, the number of white columns to be added is obtained as the round value of the difference between \( C_{w_{\text{new}}} \) and \( C_w \) (6.6), likewise the number of white rows is obtained as the round value of the difference between \( C_{h_{\text{new}}} \) and \( C_h \) (6.7).

\[ \text{Add}_\text{Cols} = \text{round}(C_{w_{\text{new}}} - C_w) \]  \hspace{1em} (6.6)
\[ \text{Add}_\text{Rows} = \text{round}(C_{h_{\text{new}}} - C_h) \]  \hspace{1em} (6.7)

Therefore

\[ R_{\text{f}} = \frac{C_w + \text{Add}_\text{Cols}}{C_h + \text{Add}_\text{Rows}} \]  \hspace{1em} (6.8)

where the \textit{round} function indicates an approximation to the above integer. The resulting ratio \( R_{\text{f}} \) generally is not equal to \( R_f \), but with the described procedure, the succeeding procedure of image reduction to the dimensions \( N C_w \) and \( N C_h \) introduces the less distortion as possible.

Hence the image is reduced to the final dimensions \( N C_h \) and \( N C_w \).

Each pixel, in the normalized image, corresponds to a rectangle of dimension \( \frac{C_{\text{new}}}{N C_w} \), \( \frac{C_{\text{new}}}{N C_h} \).
CHAPTER 6. A FONT ADAPTIVE MODULAR OCR SYSTEM

From the noise image, transformed in a black and white equivalent one, the number of black pixels of the rectangle that corresponds to a new pixel in the normalized image are calculated. The dimension in pixels of the rectangle, that corresponds to a pixel in the normalized image, is then calculated.

Such a number is variable if the dimensions of the noise image and of the normalized one are not multiple.

The level grey of each pixel is calculated as in (6.9)

\[
grey = 255 \times \frac{N_{\text{black}}}{N_{\text{tot}}} \tag{6.9}
\]

where \(N_{\text{black}}\) is the number of black pixels in the rectangle of the original image, corresponding to a pixel in the normalized one, and \(N_{\text{tot}}\) is the number of pixels in the same rectangle.

The result of the image reduction is an image of dimension \(NC_h\) and \(NC_w\) of grey level pixels, with values in the interval \([0, 255]\).

Since the values of the patterns are to be in the interval \([0, 1]\) in order to feed each neural network, a procedure of scaling the grey level values in such an interval is to be performed.

6.4 Neural architectures used for classification

Pattern classification methods considered in this work are based on neural techniques. In particular three different techniques have been tested:

1) Classification with neural networks used as autoassociators;
2) Classification derived from the combination of autoassociators and discriminating networks (Championship Algorithm);
3) Classification derived from an MLP classifier among all classes and the application of the Championship Algorithm.

6.4.1 Classification by Autoassociators

An autoassociator \(A_u\) is a particular kind of neural network which is trained by examples of only one class, to realize an identity function. Training autoassociators by examples of only one class allows each of them to perform a procedure of pattern verification [49]. Each one verifies whether a pattern belongs to the class which it is trained to identify. Let \(x_{i(l)}\) be the \(i^{th}\) input of the \(i^{th}\) neuron of the \(l^{th}\) layer. In the testing phase, for each pattern \(X\) the mean square error between the inputs and the outputs \(E_j(X) = \frac{1}{2} \sum_{i=0}^{m} (x_{i(0)} - x_{i(2)})^2\) for the autoassociators \(A_u\) related to the class \(C(u)\) is calculated, where \(x_{i(0)}\) is the \(i^{th}\) value of the input pattern \(X\), \(x_{i(2)}\) is the \(i^{th}\) value of the
output and \( m \) is the pattern dimension. Such an error indicates for a pattern the degree of belongings to the class that the autoassociator has been trained to recognize. The combination of \( N \) autoassociators \((A_0, A_1, \ldots, A_{N-1})\) equal to the number of classes can be used to perform a classification procedure.

Lower error values identify classes that the current pattern may belong to. The autoassociator with the lowest error has the highest likelihood to represent the class which the current pattern belongs to. In Fig. 6.4 is illustrated the scheme of character recognition by autoassociators only.

![Figure 6.4: Scheme of character recognition using autoassociators only.](image)

In these work each autoassociator has a topology of 84 neurons as inputs, derived from images of dimensions 7x12, as many output neurons and 20 neurons in the only hidden layer. The output of each neuron is obtained, from the activation value, by a squash-like
function.

Learning phase

Each autoassociator is trained by examples of one class only. In such a case, in order to estimate the quality of learning for an autoassociator, a technique of *cross validation* with respect to learning of the other classes can be performed.

As a matter of fact, a typical behaviour of neural networks is represented by a sort of *Uncertainty Principle* between learning and recognition. For each autoassociator there exists a set of optimal weights related to a particular learning epoch, with respect to the recognition percentage. When training phase exceeds such an epoch, learning begins to specialize on training examples, reducing generalization capability and, globally, the rate of recognition.

Every 100 epochs the training of the autoassociators is stopped and their weights are saved. At each of these steps the rate of recognition is calculated. The training is definitively stopped at the epoch that provides the maximum rate of recognition.

Classification phase

A maximum threshold value (*Rej Thres*) for autoassociator errors can be introduced to decide which is the maximum error value that a pattern can exhibit on an autoassociator in order to accept the possibility that the current pattern may belong to the related class. Otherwise the pattern is rejected. Therefore the classification or rejection criteria are based on the way the input is approximated by the output. Such a threshold is to be chosen to obtain a compromise between reliability of the entire classification system and an acceptable rejection rate.

The classification algorithm by autoassociators only is described here below.

---

**Pattern Classification by Autoassociators**

1. The current pattern *X* is presented to all the *N* autoassociators and for each one the error values

   \[ e_0 = E_0(X), \; e_1 = E_1(X), \ldots, e_{N-1} = E_{N-1}(X) \]

   are calculated, where

   \[
   E_j(X) = \frac{1}{2} \sum_{i=0}^{m} (x_{i(0)} - x_{i(2)})^2
   \]

   \[ (6.10) \]

   If
\[ \epsilon_k = \min_j \{ \epsilon_j \} \leq \text{Rej\_Thres}, \quad j \in \{0, ..., N-1\} \]

\[ \text{then} \quad X \in C(k) \]

where \( C(k) \) is the class corresponding to the \( k^{th} \) autoassociator

\[ \text{else} \quad X \text{ is rejected.} \]

\[
\begin{array}{|c|c|c|}
\hline
\text{Class} & \text{Epochs with the highest rate of recognition} & \text{Rate of recognition} \\
\hline
\text{Digits} & [350, 1400] & 99.933\% \\
\text{Lowercase letters} & [1200, 1250] & 99.740\% \\
\text{Uppercase letters} & [850, 950] & 99.043\% \\
\hline
\end{array}
\]

Table 6.1: Results of the cross-validation testing procedure for autoassociators.

**Experimental results**

The results of the cross-validation testing procedure for autoassociators of digits, lowercase letters and uppercase letters are reported in Tab. 6.1. Fixed for digits, uppercase and lowercase letters, respectively, the learning epochs which reported the best recognition results, the phase of test for autoassociators of digits, uppercase and lowercase letters are reported in Appendix A in Tabs. A.1, A.2, A.3.

The global results of recognition considering all the \( N \) classes together, at different rejection threshold in the interval \([3.0, 6.0]\), are reported in Appendix A Tab. A.4.

\( \text{Rej\_Thres} = 5.5 \) seems to be a choice that realizes a good compromise to have a low the rate of rejection and an acceptable rate of reliability.

**6.4.2 The Championship Algorithm**

The main characteristic of a classification model based on autoassociators is that it is intrinsically modular, but it is not well-suited for performing the discrimination among
very similar classes. In fact many errors in autoassociator classification derive from the short difference between the class which exhibits the lowest error and the next or the two next classes with lower errors. In order to recover such errors a set of discriminating networks between two classes are introduced. Such networks, with MLP architecture and classifier structure [50] [51], are aimed at discriminating very confused classes. We denote a single discriminating network between class \( C(u) \) and class \( C(v) \) as \( D_{u,v} \), where \( u < v, \ u, v \in \mathbb{N} \).

The use of discriminating networks between two classes is justified by the fact that such networks, trained to discriminate between a pair of classes, exhibit better performances than a network aimed at discriminating more than two classes.

First, the current pattern is presented to all the autoassociators and the related classes are ordered on the basis of the input–output error. \( \varepsilon_{q(j)} \) represents the error of the pattern with respect to the class \( i^{th} \), where \( i = q(j) \), and \( q \) is a permutation of \( i \) such that \( \varepsilon_{q(j)-1} \leq \varepsilon_{q(j)} \), \( j = 1, \ldots, N - 1 \). Hence, the first \( N_{\text{champ}} \) classes (\( 1 < N_{\text{champ}} \leq N \), where \( N \) is the total number of classes) are considered. The current pattern is presented to the discriminating networks, related to the first \( N_{\text{champ}} \) classes, if \( |\varepsilon_{q(0)} - \varepsilon_{q(j)}| \leq d, \ 0 < j < N_{\text{champ}} \), where \( d \) is a fixed upper threshold. To the winning class of each competition is given a score. The competition between each of the couples of classes is extended to all the \( N_{\text{champ}} \) classes considered by reporting the final score of the single matches in a ranking list. Basically, the championship winner is selected among the first \( N_{\text{champ}} \) classes obtained by autoassociators, using a competition that resembles a football championship.

The championship algorithm is shown in Fig. 6.5.

---

**Pattern Classification by Championship Algorithm**

1. The current pattern \( X \) is presented to all the \( N \) autoassociators and for each one the error values, \( \varepsilon_0 = E_0(X), \varepsilon_1 = E_1(X), \ldots, \varepsilon_{N-1} = E_{N-1}(X) \), are calculated.

2. Errors \( \varepsilon_i, 0 \leq i \leq N - 1 \) are disposed in ascending order (\( \varepsilon_{q(0)} \leq \cdots \leq \varepsilon_{q(N-1)} \)), where \( i = q(j) \) (a permutation of \( i \) so that \( \varepsilon_{q(j-1)} \leq \varepsilon_{q(j)} \)) is the index of the class).

3. If \( \varepsilon_{q(0)} \geq \text{Rej.Thres} \) then

   \( X \) is rejected

   else
6.4. NEURAL ARCHITECTURES USED FOR CLASSIFICATION

Figure 6.5: Scheme of character recognition using the Championship Algorithm.

if

$$|\epsilon_{q(0)} - \epsilon_{q(j)}| \leq d \quad 1 \leq j < N_{\text{champ}};$$

then

the current pattern is presented to each discriminating network $D_{q(j), q(k)}$, $0 \leq j < k < N_{\text{champ}}$. Each winning class in each match is given a score of

$$2 \cdot \max_{i \in \{0,1\}} \{x_{i(2)}\}$$

(6.14)

where $x_{i(2)}$ represents the output value of the $i^{th}$ neuron of the current discriminating network.

The class with the highest score is assigned to the current pattern.

else
The use of a score, weighted with respect to the maximum value among outputs is introduced in order to avoid the possibility, although remote, that in every match a different winning class is obtained.

Moreover, in this approach, the ranking of autoassociator errors becomes only a set of hypotheses of classification with decreasing likelihood. The final decision is postponed and is carried out by the set of discriminating networks.

The integral application of the *Championship Algorithm* results computationally expensive, both in the learning and in the recognition phase, when dealing with a high number $N$ of classes. In this case, during the learning phase, a number of $N$ autoassociators equal to the number of classes and $\binom{N}{2}$ discriminating networks should to be trained. As preliminary procedure only the networks for the very confused classes are trained. The other networks are trained automatically at the purpose in the testing phase of the system.

In order to reduce the computational cost in the recognition step, after having calculated the error for each of the $N$ classes using autoassociators, as discussed, only $\binom{N_{\text{champ}}}{2}$ further forward steps related to as many discriminating networks are performed, where $1 < N_{\text{champ}} \leq N$ is the number of classes that take part to the championship. In this work th value $N_{\text{champ}} = 2$ is used, therefore only $\binom{2}{2} = 1$ further forward step is performed. In our case, this number is an upper bound for forward steps of discriminating networks. For example for $N_{\text{champ}} = 2$, if the condition on $d$ is not satisfied the discriminating network is not used.

### Experimental results

Using autoassociators of the previous experiments, with $\text{Rej}_\text{Thres} = 5.5$, and some discriminating networks for the very confused classes, for variable values of the parameter $d$ in the interval $[0.1, 0.5]$, the *Championship Algorithm* has produced the results in Appendix B Tab. B.4. Being $\text{Rej}_\text{Thres} = 5.5$, the best results are obtained with $d = 0.2$ both in terms of rate of recognition and of rate of reliability.

The best results of the *Championship Algorithm* for each single set of characters (digits, uppercase and lowercase letters) are reported in Appendix B, Tabs. B.1, B.2, B.3.
6.4. NEURAL ARCHITECTURES USED FOR CLASSIFICATION

Table 6.2: Computational cost for autoassociators and for the Championship Algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Learning procedures</th>
<th>Forward steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoassociators</td>
<td>$N$</td>
<td>$N$</td>
</tr>
<tr>
<td>Championship Algorithm</td>
<td>$N + \binom{N}{2}$</td>
<td>$N + \binom{N_{champ}}{2}$</td>
</tr>
</tbody>
</table>

6.4.3 MLP Classifier and Championship Algorithm

As discussed in § 6.4.2, the use of autoassociators or autoassociators combined with discriminating networks (Championship Algorithm) could be computationally expensive with respect to the use of a single network, with MLP architecture and classifier structure, among $N$ classes. Being $N$ the classes to be considered, the computational cost in terms of number of learning procedures and number of forward steps are reported in Tab. 6.2.

In Tab. 6.2 $N_{champ}$ is the number of classes that take part to the championship.

In order to reduce the computational cost in the recognition step, that may be considerable when $N$ grows, it has been experimented a technique of classification that provides a preliminary level of classification represented by a network, with MLP architecture and classifier structure, among all the $N$ classes of the problem. Such a technique is only intended to speed up the recognition phase by reducing the number of forward steps, preserving the rate of recognition obtained by the Championship Algorithm. Such a rate is an upper bound for this technique.

First of all, a pattern is presented to the MLP network that classifies the patterns among $N$ classes, and the first $N_{mlp}$ classes, $N_{champ} \leq N_{mlp} \leq N$, that have the higher outputs are selected. Then the pattern will be presented to the autoassociators related to the $N_{mlp}$ classes. Among the $N_{mlp}$ classes selected by the MLP classifier, autoassociators select the $N_{champ}$ classes which take part, in a second step, to the championship.

The scheme of such technique is shown in Fig. 6.6.

The number of $N_{mlp}$ classes selected by the MLP network is not a fixed value. It derives from the number of output neurons, whose output levels are greater or equal than a minimum value $Out_{Thres}$, that is calculated as a percentage $p$ of the maximum output value, $x_{\max}(i) = \max\{x_i(l)\}$, $i = 0, ..., N - 1$, of the output level neurons ($l = 2$), as it is illustrated in (6.15).

$$Out_{Thres} = x_{\max}(l) * (1 - p), \quad l = 2 \quad (6.15)$$

Therefore the $N_{mlp}$ classes that are to be presented to the autoassociators are given by (6.16)
Figure 6.6: Scheme of character recognition using an MLP classifier combined with the Championship Algorithm.

\[ N_{mlp} = |\{x_i(2) \in [\text{Out}_\text{Thres}, x_{max}(2)]\}| \quad (6.16) \]

**Pattern Classification by MLP + Championship Algorithm**

1. The current pattern \( X \) is presented to the MLP network.
2. The class which corresponds to the output \( x_{max}(2) \) of MLP, where \( x_{max}(2) = \max_i \{x_i(2)\}, \ i = 0, \ldots, N - 1 \), is selected.
3. The number \( N_{mlp} \) of the classes, for which (6.15) and (6.16) hold, are selected.
4. The *Championship Algorithm* for the selected \( N_{mlp} \) classes is performed.

This procedure reduces the computational cost in the recognition step. The cost is increased of one learning procedure and of one forward step related to the MLP classifier among \( N \) classes, but the number of forward steps for autoassociators is reduced at \( N_{mlp} \leq \).
6.5. THE SEGMENTATION PROCEDURE OF THE ON-LINE PHASE

$N$. Globally the computational cost of such a procedure is reported in Tab. 6.3, being $N_{champ} \leq N_{mlp} \leq N$.

<table>
<thead>
<tr>
<th>Learning procedures</th>
<th>Forward steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1+N+(\frac{N}{2})$</td>
<td>$1+N_{mlp}+(\frac{N_{champ}}{2})$</td>
</tr>
</tbody>
</table>

Table 6.3: Computational cost of the MLP classifier and of the Championship Algorithm.

**Experimental results**

The classification procedure that uses the combination of an MLP classifier among $N$ classes and the Championship Algorithm has reported the results illustrates in Appendix C Tab. C.1. The neural architecture has been tested on 26234 patterns with variable values of the parameter $p$ in the interval $[0.996, 0.999]$.

**6.5 The segmentation procedure of the on-line phase**

Characters segmentation is a decisive procedure in order to obtain a high rate of recognition during the on-line recognition phase. In fact, providing the classification system with character images which keep the most of their features, without including features belonging to near characters, makes classification procedure more robust.

Some of the methods described in § 6.1 result quite robust from this point of view. For example, dissection methods or recognition-based segmentation ones, in theory, can maintain the most of character features if the right positions of real or virtual segmentation are located. However, it is not easy to locate such a point, with the result of obtaining segmentations with loss or corruption of information.

In this work an iterative connected component extraction procedure, by adaptive binarization threshold, is used. The segmentation of characters in a image that contains characters can be performed by a procedure of connected component extraction. Then the average width of the connected components is calculated. Then the clustering of connected components into words and words into rows can be performed. Such a procedure, during document processing, can also be considered as part of the document analysis phase. The components whose width is greater than a threshold, that depends on the average width of components, are submitted to an iterative procedure of connected component extraction, tuning the binarization threshold towards more selective values, until
the dissection of the component into components of width below the average width is obtained.

Since printed characters result rarely hard connected, this procedure appears well-suited for character dissection in this environment, without significant loss of information. In fact, the loss of information due to the use of a more selective value of binarization, is balanced by the fact that the first and the last characters of a set of connected characters present at least on certain point of dissection, represented by the left and the right limit of the component.

The input of the system is a grey scale image, whose pixels have grey-level values in the interval [0, 255]. As discussed in § 6.2, the lowest grey level (0) corresponds to white pixels, while the higher (255) corresponds to black ones. Such a representation provides more information than an equivalent black and white one with the same resolution.

The segmentation procedure is composed by the following phases:

1. The system performs a binarization of the image with a low threshold ($Bin_{Thres} = 100$) in order not to loose significant features of each character. The result is a black and white image (binary image) with only two grey levels (0 = white, 255 = black);

2. a procedure of extraction of connected components is performed. Such a procedure allows to separate not touching characters from the binary image;

3. the average width ($W_{av}$) and height ($H_{av}$) of the components is calculated;

4. the component extracted are ordered with respect to the barycentre increasing abscissa;

5. an iterative procedure allows to cluster two components if the distance $L$ between the points $(x_{max}(i), y_{max}(i))$ and $(x_{min}(j), y_{max}(j))$ of the components $i^{th}$ and $j^{th}$, respectively, is a segment with an angle in the interval $(-75^\circ, +75^\circ)$ and length $L$, $w_{min} < L < W_{av} \cdot (1 + \epsilon_d)$ (Fig. 6.7) where $w_{min}$ is a heuristic value, or simply $L \leq w_{min}$, without any further condition for the angle if the two components are very close;

6. word components are clustered with respect to the values of the ordinates. Such a procedure is performed in a iterative fashion: word components with barycentre ordinate value in a range $[(y(i) - \epsilon_h), (y(i) + \epsilon_h)]$, $\epsilon_h = \frac{H_{av}}{2}$, with respect to the barycentre ordinate value $y(i)$ of the $i^{th}$ component as reference, will belong to the same text row;

7. if a component has a width $W_c < W_{av}(1 + \epsilon_d)$, $0 \leq \epsilon_d < 1$ the probability that the component represents a single character is high and the system may classify such a component, or may reject it if it is not able to express a reliable pronunciation.
Figure 6.7: *Conditions for clustering components into words.*

Otherwise the system tunes the binarization threshold towards more selective values (in particular it is increased as $Bin_{Thres} = Bin_{Thres} + 10$) and performs a new connected component procedure in the sub-image of the component. Such a procedure is iteratively repeated till the old component is decomposed in one or more components for whose width $W_c$ holds $W_c \leq W_{av}(1 + \epsilon_d)$.

The procedure of classification of each single component obtained by such iterative segmentation is newly performed. Such a procedure of iterative segmentation and classification is carried on till all components of each word of each text row are classified or rejected.

In Fig. 6.8 an example of iterative segmentation is shown.
Figure 6.8: Example of iterative segmentation by adaptive threshold.


Chapter 7

Conclusions

Paper is still the main medium for exchanging information in literary, scientific or commercial field. However the need of sharing information in electronic format, as well as of creating databases of data extracted from documents, in order to compress them and handle them in a more selective fashion, has increased the demand of tools to automatically convert information hold on paper into digital one.

Many architectures have been proposed in order to solve such a problem. They can be grouped into three different types.

The first group is represented by architectures intended to process a single kind of documents having a fixed logical and physical structure [1], [2], [4]. They are characterized by a high degree of reliability but they lack flexibility.

The second one is represented by systems which use a knowledge-based approach to document processing. The knowledge is used directly in the understanding phase without a predefined goal of reading [5], or in order to provide a probabilistic attribution of a semantic label to the extracted objects [17], [6], [7]. Such systems are characterized by a great flexibility, but they lack reliability.

The third one is represented by systems intended to realize a compromise between reliability and flexibility. These systems aim at processing documents of a domain that can be divided into classes. Such systems are intended to individuate the class a document belongs to, and to use a proper model associated to the recognized class that allows to understand the current document. Such systems may provide also the possibility of constructing a new model for an unknown class. This possibility is usually dwelled upon the user who describes the model for a new class by locating the physical objects and providing them with a semantic label [9]. Otherwise the construction is performed by machine learning techniques [12], [13], [11], but it leads to a probabilistic semantic attribution of the objects.

In the present work, a general architecture for processing documents belonging to
a domain which can be partitioned in classes, characterized by the invariability of the
position of the objects in terms of coordinates, has been presented. The system is based
on a general knowledge, formalized by a semantic network which describes the document
domain under consideration. Such a knowledge attempts to capture some structural and
logical similarities among the classes by describing logical relationships among document
parts and their physical constraints.

The knowledge on domain has been used as a procedural scheme for processing doc-
uments of a known class, or for a semi-automatic construction of a specific model for an
unknown class of the domain of interest. Human interaction for the document model con-
struction is provided only in confirmation or refutation of a semantic label suggested by
the system for each physical extracted object.

For the characteristics of the classes and of the related models, the system performs a
reliable labelling of the extracted physical objects. This is due also to the introduction of
a phase of document classification that can greatly simplify the phase of understanding.
In fact it limits the variability of the possible layout and logical structures a document of a
class may assume, with respect to the logical-layout relationships allowable for a document
in the whole domain. To reach a reliability which is comparable to the one of the systems
designed for a single class of documents, the system has been designed to process classes
of documents characterized by a fixed physical and logical structure.

An application of the proposed architecture has been presented. It refers to the invoice
domain, which is a typical domain of documents divided into classes [39], [52], [53], [36],
[37].

The architecture allows to obtain a document processing system characterized by a
degree of reliability, that is comparable with the reliability of systems oriented to process
a single class of documents. On the other hand, it allows to obtain a certain flexibility in
the domain of interest at least. The general knowledge could also be used directly for the
understanding phase of different classes of documents, accepting a loss of reliability. Such
a framework can be adopted for every domain of documents divided into classes, where a
module of class recognition can be designed.

In order to extract the information from the labelled objects an OCR system, developed
with neural techniques, has been proposed [54]. It is oriented to recognize digits, lowercase
and uppercase letters. In particular the comparison among three kinds of neural-based
classification methods has been presented. The classification method derived from a hier-
archical combination of autoassociators and discriminating networks between two classes
(*Championship Algorithm*) has exhibited better performances with respect to the classifi-
cation by autoassociators only. Since the OCR system is concerned with a large number of
classes, the classification procedure results computationally expensive. Therefore an MLP
classifier is used in order to provide a preliminary selection of the most likely classes. This
preliminary selection reduces the computational cost of the classification step, limiting
the number of neural forward phases at the autoassociator level, but it may degrade the
rate of reliability. Therefore, some experiments intended to reduce the computational cost limiting the reduction of the recognition rate have been presented. A possible direction for future researches can be represented by training the MLP classifier or a clustering network as a radial basis function one, in order to provide a better selection of likely classes, so that the whole system could reach the rate of reliability of the *Championship Algorithm*.

Moreover a technique for character segmentation, as a phase of document processing, based on the recurrent extraction of connected components by an adaptive binarization threshold, has been proposed.

The whole architecture is being implemented within an Esprit project.
Appendix A

Experimental results on character recognition using autoassociators

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>489</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>492</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>372</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>304</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>304</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>468</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>468</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>462</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>462</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>435</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>435</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>432</td>
<td>0</td>
<td>0</td>
<td>432</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>465</td>
<td>0</td>
<td>0</td>
<td>465</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>474</td>
<td>0</td>
<td>474</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>444</td>
<td>444</td>
</tr>
</tbody>
</table>

Recognition rate  99.933%

Table A.1: Confusion Table for digits. Learning epoch = 1250.
### Table A.2: Confusion Table for uppercase letters. Learning epoch = 900.

|          | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Z | Pat |
| **A**    | 152 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **B**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **E**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **F**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **G**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **H**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **I**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **J**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 486 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **K**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 486 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **L**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 486 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **M**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 486 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **N**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **O**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **P**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Q**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **R**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **S**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **T**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **U**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **V**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **W**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **X**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Z**    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Recognition rate 99.043%
Table A.3: Confusion Table for lowercase letters. Learning epoch = 1250.

| a | b | c | d | e | f | g | h | i | k | l | m | n | o | p | r | s | t | u | v | Pat |
| a | 448 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 448 |
| b | 0 | 495 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 495 |
| c | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| d | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| e | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| f | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| g | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| h | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| i | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| k | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| l | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| m | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| o | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| p | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 496 |
| r | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 496 |
| s | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 496 |
| t | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 496 |
| u | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 496 |
| v | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 496 |

Recognition rate 99.740%
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Rate of recognition</th>
<th>Rate of error</th>
<th>Rate of rejection</th>
<th>Rate of reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Autoassociators</strong> Rej_Thres = 3.0</td>
<td>96.834%</td>
<td>1.835%</td>
<td>1.331%</td>
<td>98.140%</td>
</tr>
<tr>
<td>for all the N classes</td>
<td>Number of patterns = 26234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Autoassociators</strong> Rej_Thres = 3.5</td>
<td>97.444%</td>
<td>1.914%</td>
<td>0.642%</td>
<td>98.074%</td>
</tr>
<tr>
<td>for all the N classes</td>
<td>Number of patterns = 26234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Autoassociators</strong> Rej_Thres = 4.0</td>
<td>97.789%</td>
<td>1.997%</td>
<td>0.214%</td>
<td>97.998%</td>
</tr>
<tr>
<td>for all the N classes</td>
<td>Number of patterns = 26234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Autoassociators</strong> Rej_Thres = 4.5</td>
<td>97.856%</td>
<td>2.057%</td>
<td>0.087%</td>
<td>97.942%</td>
</tr>
<tr>
<td>for all the N classes</td>
<td>Number of patterns = 26234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Autoassociators</strong> Rej_Thres = 5.0</td>
<td>97.916%</td>
<td>2.057%</td>
<td>0.028%</td>
<td>97.943%</td>
</tr>
<tr>
<td>for all the N classes</td>
<td>Number of patterns = 26234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Autoassociators</strong> Rej_Thres = 5.5</td>
<td>97.916%</td>
<td>2.057%</td>
<td>0.028%</td>
<td>97.943%</td>
</tr>
<tr>
<td>for all the N classes</td>
<td>Number of patterns = 26234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Autoassociators</strong> Rej_Thres = 6.0</td>
<td>97.916%</td>
<td>2.057%</td>
<td>0.028%</td>
<td>97.943%</td>
</tr>
<tr>
<td>for all the N classes</td>
<td>Number of patterns = 26234</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.4: Experimental results for all classes, using autoassociators.
Appendix B

Experimental results on character recognition using the Championship Algorithm

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>rej</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>462</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>372</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>404</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>404</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>468</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>468</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>462</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>462</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>465</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>465</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>432</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>432</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>465</td>
<td>0</td>
<td>0</td>
<td>465</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>444</td>
<td>0</td>
<td>444</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>444</td>
</tr>
</tbody>
</table>

Recognition rate 99.933%
Error rate 0.067%
Rejection rate 0.000%
Reliability rate 99.933%

Table B.1: Confusion Table for digits: rej\_Thres = 5.4, d = 0.4.
# APPENDIX B. EXPERIMENTS ON CHAMPIONSHIP ALGORITHM

## Table B.2: Confusion Table uppercase letters: \(\text{rej}_\text{Thres} = 5.4, \ d = 1.2\).

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>I</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>429</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>446</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>434</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>494</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>560</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>483</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>498</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>496</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>487</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>490</td>
</tr>
<tr>
<td>O</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>O</th>
<th>F</th>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>U</th>
<th>V</th>
<th>W</th>
<th>X</th>
<th>Z</th>
<th>Pat</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>432</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>500</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>477</td>
</tr>
<tr>
<td>D</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>498</td>
</tr>
<tr>
<td>O</td>
<td>490</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>483</td>
</tr>
<tr>
<td>P</td>
<td>0</td>
<td>494</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>494</td>
</tr>
<tr>
<td>Q</td>
<td>4</td>
<td>0</td>
<td>494</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>494</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>477</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>495</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>477</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>494</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>494</td>
</tr>
<tr>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>495</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>496</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>497</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>498</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>490</td>
</tr>
</tbody>
</table>

| Recognition rate | 99.544% |
| Error rate       | 99.376% |
| Rejection rate   | 0.081%  |
| Reliability rate | 99.624% |
|   | a | b | c | d | e | f | g | h | i | k | l | m | n | o | p | r | s | t | a | c | rej | rct |
| a | 448 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 448 |
| b | 495 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 495 |
| c | 0 | 416 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 416 |
| d | 0 | 0 | 456 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 456 |
| e | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| f | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| g | 0 | 0 | 0 | 0 | 0 | 486 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 486 |
| h | 0 | 0 | 0 | 0 | 0 | 0 | 500 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 500 |
| i | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 488 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 488 |
| k | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| l | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| m | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 500 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 500 |
| n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 488 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 488 |
| o | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 496 |
| p | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 500 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 500 |
| q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 500 | 0 | 0 | 0 | 0 | 0 | 0 | 500 |

Table B.3: **Confusion Table lowercase letters**: $\text{rej}_\text{Thres} = 4.6$, $d = 0.8$. 

<table>
<thead>
<tr>
<th>Recognition rate</th>
<th>99.948%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate</td>
<td>0.052%</td>
</tr>
<tr>
<td>Rejection rate</td>
<td>0.000%</td>
</tr>
<tr>
<td>Reliability rate</td>
<td>99.948%</td>
</tr>
</tbody>
</table>
### APPENDIX B. EXPERIMENTS ON CHAMPIONSHIP ALGORITHM

#### Parameters

<table>
<thead>
<tr>
<th>Championship Algorithm between the first two classes of the autoassociators ranking list</th>
<th>Rej. Thres = 5.5 d = 0.1 Number of patterns = 26234</th>
<th>Rate of recognition</th>
<th>Rate of error</th>
<th>Rate of rejection</th>
<th>Rate of reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>97.935%</td>
<td>2.037%</td>
<td>0.028%</td>
<td>97.963%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>97.963%</td>
<td>2.009%</td>
<td>0.028%</td>
<td>97.990%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>98.019%</td>
<td>1.954%</td>
<td>0.028%</td>
<td>98.046%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>97.924%</td>
<td>2.049%</td>
<td>0.028%</td>
<td>97.951%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>97.967%</td>
<td>2.005%</td>
<td>0.028%</td>
<td>97.994%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>97.678%</td>
<td>2.294%</td>
<td>0.028%</td>
<td>97.705%</td>
</tr>
</tbody>
</table>

#### Table B.4: Experimental results for all classes, using the Championship Algorithm.
Appendix C

Experimental results on character recognition using MLP and Championship Algorithm
### Table C.1: Experimental results for all classes, using MLP classifier combined with the Championship Algorithm.

<table>
<thead>
<tr>
<th>MLP + Championship Algorithm</th>
<th>p = 0.9996</th>
<th>Rate of recognition</th>
<th>96.644%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rej_Thres = 5.0</td>
<td>Rate of error</td>
<td>3.107%</td>
</tr>
<tr>
<td></td>
<td>d = 0.2</td>
<td>Rate of rejection</td>
<td>0.250%</td>
</tr>
<tr>
<td></td>
<td>Number of patterns = 26234</td>
<td>Rate of reliability</td>
<td>96.885%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average number of autoassociators involved</td>
<td>14.256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MLP + Championship Algorithm</th>
<th>p = 0.9997</th>
<th>Rate of recognition</th>
<th>96.465%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rej_Thres = 5.0</td>
<td>Rate of error</td>
<td>3.321%</td>
</tr>
<tr>
<td></td>
<td>d = 0.2</td>
<td>Rate of rejection</td>
<td>0.214%</td>
</tr>
<tr>
<td></td>
<td>Number of patterns = 26234</td>
<td>Rate of reliability</td>
<td>96.672%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average number of autoassociators involved</td>
<td>15.443</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MLP + Championship Algorithm</th>
<th>p = 0.9998</th>
<th>Rate of recognition</th>
<th>96.858%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rej_Thres = 5.0</td>
<td>Rate of error</td>
<td>2.984%</td>
</tr>
<tr>
<td></td>
<td>d = 0.2</td>
<td>Rate of rejection</td>
<td>0.159%</td>
</tr>
<tr>
<td></td>
<td>Number of patterns = 26234</td>
<td>Rate of reliability</td>
<td>97.011%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average number of autoassociators involved</td>
<td>17.254</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MLP + Championship Algorithm</th>
<th>p = 0.9999</th>
<th>Rate of recognition</th>
<th>96.976%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rej_Thres = 5.0</td>
<td>Rate of error</td>
<td>2.877%</td>
</tr>
<tr>
<td></td>
<td>d = 0.2</td>
<td>Rate of rejection</td>
<td>0.147%</td>
</tr>
<tr>
<td></td>
<td>Number of patterns = 26234</td>
<td>Rate of reliability</td>
<td>97.119%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average number of autoassociators involved</td>
<td>20.182</td>
</tr>
</tbody>
</table>
Bibliography


