Managing Multiligual OCR Project using XML

Gaurav Harit[1], K. J. Jinesh[2], Ritu Garg[3] C.V Jawahar[2], and Santanu Chaudhury[3]

[1] Indian Institute of Technology, Kharagpur
[2] International Institute of Information Technology, Hyderabad
[3] Indian Institute of Technology, Delhi

Abstract. This paper presents a scheme based upon XML based labeling for managing a large multilingual OCR project. Managing a large multi-lingual OCR project involving multiple research groups, developing script specific and script independent technologies in a collaborative fashion is a challenging problem. In this paper, we present some of the software and data management strategies designed for the project aimed at developing OCR for 11 scripts of Indian origin for which mature OCR technology was not available.

1 Introduction

Managing a large multi-lingual OCR project involving multiple research groups, developing script specific and script independent technologies in a collaborative fashion is a challenging problem. In this paper, we present some of the software and data management strategies designed for the project aimed at developing OCR for 11 scripts of Indian origin for which mature OCR technology was not available. The challenges involved in the project management were the following:

1. Generation and management of ground truth for different scripts which can serve as benchmark for evaluation of OCRs since no such benchmark data set exist for Indian scripts.
2. Standardization of the representation scheme for the processed document image data at intermediate stages so that different modules can be independently developed and composed in an application/script specific fashion.
3. Devising a representation mechanism for deployment of script specific OCR for all or part of the document page for conversion to electronic format for a multi-lingual multi-script environment.
4. Representation scheme for the final OCRed output so that it can be rendered electronically preserving the layout amenable for electronic editing, searching and browsing.
5. Representation of the content that can facilitate semantic annotation, and use within a conceptual retrieval engine.

We devised an XML based tagging structure for meeting these challenges. In that sense, we devised our tagging scheme with more generic objectives than those of similar approaches [1–3]. We preferred an XML based scheme over

HTML based approaches because of the generality of the tagging structure, and the flexibility of the tagging scheme to accommodate semantic annotation for conceptual search in the realm of multi-lingual semantic web. The XML based scheme enabled us to organize document images not as raw images or unstructured Unicode text blocks, but as a collection of logically related components. Obviously success of the representation scheme depends upon quality of the output obtained from the image analysis algorithms and OCR engines.

Addressing representational issues for effective data representation and efficient information flow has been one of the central research thread in multimedia and multilingual document analysis. An early attempt for a standard data exchange format came from the UNIPEN [2, 3] for representing online handwritten information. Most of the representational schemes employed some form of a tagged representation as in [1], where the authors employed a HTML/XHTML representation. As the expectations out of a standard representation grew, it was relaized that an XML based representation is better suited for multimedia and document analysis tasks [4]. This makes it suitable for a spectrum of diverse applications, while being compatible with futuristic extensions. The UPX format [5], a successor to UNIPEN and all its derivatives or extensions [6] use XML for the basic data representation. UPX format is based on W3C developed InkML [7]. For printed document analysis, XML has been used earlier for tasks such as representation of annotated data sets [8, 9]. Specific problems related to developing and representing datasets for Indian language OCR research is addressed in [9].

A hard/rigid top-down software architecture is often unsuitable for a dynamic research project. The software frame work needs to provide enough flexibility for the individuals to experiment, while working towards a common goal. The problem is more severe when the project involve multiple organizations. XML based interfaces across various modules of the character recognition system makes it easy to develop, integrate and test in a seemless manner. An XML based framework was proposed [10] recently for projects in cognetive vision domain for wasier collaboration and distribution of work.

Semantic web technologies [11] also uses XML as their underlying data representation and exchange format. Lot of efforts already went into effective use of XML for multimedia datasets [12, 13]. Use of XML as a standard format for cross-platform integration was being explored for a long time [14]. It grew up as an industry standard in the past decade. The underlying representation we use for our project management and data representation is XML based. Our representation is motivated by the need for a solution which addresses the challenges listed in Section 2. We briefly describe our solution in Sections 3, 4, 5 and 6.

2 Challenges in Indian Multilingual Context

Indian multilingual context introduces a variety of challenges to the research, design, development and validation of character recognition systems. Here we brifly describe some of them. Eighth schedule of Indian constitution have a list

of 22 official languages. Apart from these, there are numerous other languages and dialects under circulation. Some of these languages share the same script (for example, Bengali, Bishnupriya Manipuri, Assamese use almost the same script). Most of the Indian languages have originated from a couple of common languages like Brahmi.

Common Ancestry of Languages The common ancestry of the Indian languages may seem to be a great advantage for developing character recognizers. However, these languages, though share the common alphabet, differ significantly in the script. The script used in northern part of India is structurally very different from that of south Indian languages. And even within a region, shape of gliffs and their 2D distribution,to form the script, vary significantly.

Collaborative and Distributed Research Activities For the sucessful development of multilingual Indian OCRs, we need active involvement from people who are geographically separated and with widely varying expertise. This necessitated a development process which is designed for seemless integration modules which are developed distributedly. Significant amount of flexibility is required for designing workflows which suits varying requirements. This requires efficient, stable and consistant communication framework between modules developed by each of the participants to achieve the desired targets.

Encoding and Representation Most of the Indian scripts are not completely compatible with the widely accepted global standards such as UNICODE. This has raised large number of issues of practical interest, which are still unsolved Moreover in late 1960s and early 1970s, the script of many languages were truncated for fitting the language into the type writer and electronic typesetting. This ended up in a mixed symbol set for many languages, without a standard. Our representations, while supporting unicode, are also compatible with other parallel attempts and popular representations.

Representation and Recognition Characters in the Indian languages change the appearence (shape) based on the context. Vowels and rarely consonants form *matras* when combined with other consonants. When two or more consonants are combined, Indian Languages show a tendency to form conjunct symbols. Latin scirpts have characters as their basic building blocks. The smallest entity that can be easily extracted from a document in such language is the charcter. In Indian languages, syllables form the basic building blocks. However, pattern classification is practically impossible at the sylable level due to the difficulty in the reliability of segmentation. Also the number of distict sylables (few thounsands) could be very high in a typical Indian language. Our internal and intermediate representations of the OCRs seemlessly allow simultaneous representations of words, *akshara*, unicode, and symbols.

Performance Evaluation Performance evaluation of OCRs and related technologies require a standard data set and a consistant interface. There should a significantly large and diverse benchmark data to make the performance

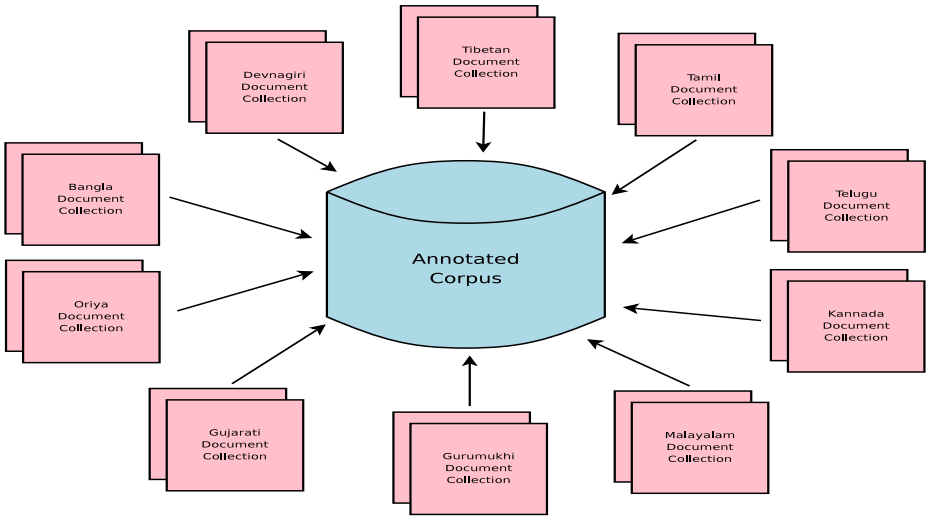

Fig. 1. Database Architecture for the Multilingual Annotated Database.

evaluation statistically significant. This require a common representational schema suiting a variety of document iamge catagories, for data representation and access. To make the data accessible to diverse users/applications, access methods with enogh flexibility is required. We achieve both these with the help of an XML representation.

Developing Multilingual Applications A consistant single framework for communication across all the associated modules, will ensure ease in development. Same will ensure easier enhancement of the system. Multilingual application requires communication between different modules, which does processing related to different scripts and methods. A consistent, standard interface will ensure easier conversion of research outcomes to industry standard commercial products.

3 Ground Truth Management

Supervised machine learning algorithms require ground truth data for learning. The effectiveness of algorithm depends largely on the data on which it is trained. Like in case of handwriting systems [6, 5] effective use of XML for storage of data as demonstrated in [8, 9] is chosen for storage of the annotation. It will make the semantic relationship between the components exploitable for developing user-friendly access modalities for these document images. The database comprise of scanned document images of different Indian scripts. Since the entire collection include different language document under one roof (database), a well defined methodical annotation scheme has been developed. The database architecture

we follow, is shown in Figure 1. Annotated Corpus provide the data annotated at functional, structural and content levels.

Corpus of Images and Text: A large collection of document images is required for development and testing of many document image analysis algorithms. In addition to the raw document image corpus, a parallel pre-processed image collection is maintained. Parallelism of data will be maintained by appropriate directory structure and file naming conventions as in Figure 2. A corpus of the text corresponding to a set of document images is prepared. In addition to the above image and text corpus, document image suited for various other tasks (like text-graphics separation) is also prepared with corresponding annotation. All meta details of the scanned documents are stored as in [9].

Content level annotation and access: For evaluation of the OCR accuracies, we need alignment of textual content with that of image segments. Annotated corpus will contain different hierarchy of segmentation on images and its corresponding content [8]. APIs are provided for C/C++ to effectively use the annotated data with OCRs for training and evaluation. Annotation is done with the help of a specially designed tool [9]. The annotation schema is broadly classified into four categories based on the type of meta-information and annotation data to be stored.Figure 3 shows the block diagram of the schema.

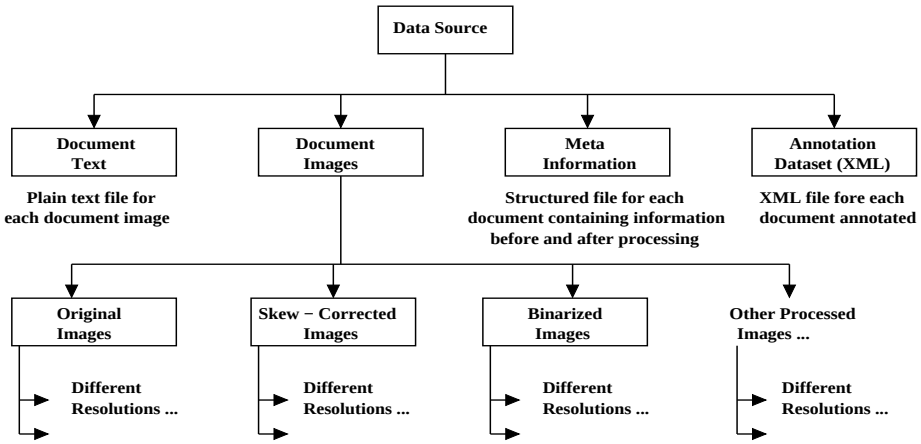

Fig. 2. Hierarchical storage of document image and text content in a directory structure

Annotation Schema: Large datasets required for OCR development and evaluation call for a standard representation that is independent of scripts and allows semantic interpretation of the data at various user-defined logical levels. Hierarchy of XML tags is a nice solution in that front. Annotation has meta-information

of the sources along with the actual data. The process starts with annotation of the document images at block level. That is, segmenting the image into different blocks and labeling them as text and non-text blocks. The second level of annotation is for lines and then it continue down to words and *Aksharas*. The non-text parts of the document image will also be annotated at higher level like, graphics, table, picture etc.

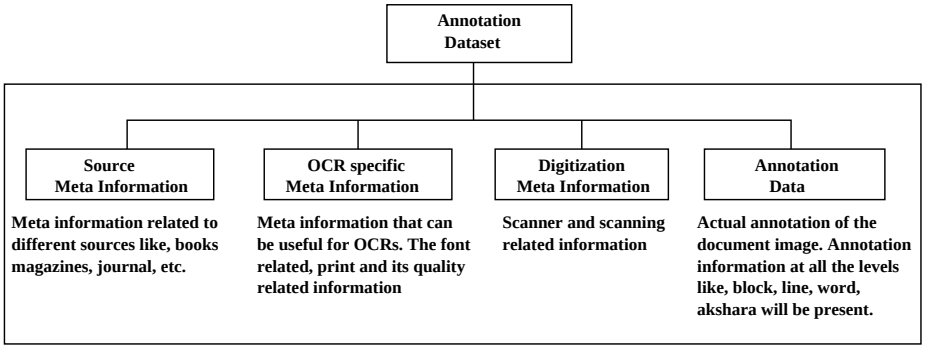

Fig. 3. The document image meta information and annotation storage elements and sub-elements.

Sources meta-information: The general information about the sources of the document page is stored as meta data. The sources of document page can be classified into five categories. Each of the categories has specific representation in the schema. All have some common attributes. (i) Unique ID; (ii) Name or title; (iii) Name of Publisher or owner; (iv) Date of Publication; (v) Language of Publication; Books and Magazines/Journals store the *Subject* information. While Both Magazine/Journal and Newspapers store their *place of publication*. Other than these, for books the following informations are also stored,*Name of the books author/editor*,*Bar code*,*ISBN code* and *Edition number*. In case of magazines we store *Volume information* of Magazines, while *License Number* is stored in case of newspaper.

OCR specific meta-information: We store a qualitative description of the qality of the source. We define four quality levels. (i) QA: Very good quality print like, laser print on a clean paper with resolution greater than 300dpi. (ii) QB: Good quality print on ordinary paper with resolution around 100-300dpi. (iii) QC: Average quality print on any ordinary paper. Type writer prints etc. (iv) QD: Bad quality print on low quality paper and low quality ink in which the back side characters are also visible along with noise, cuts and breaks etc. *Type settings* used in printing of the source. *Font type and size* specific information. All of the above information may not be available in the source itself. An expert may have to predict some of the attribute values. In that case, based on the number of

attributes mentioned in the source and predicted by an expert a *reliability factor* has to be specified for the meta-information.

Digitization meta-information: Document images are obtained from the above mentioned sources using different types of instruments under different settings. This information may be useful in applying specific type of algorithms in annotation and obtaining correct annotation. (i) Type and properties of scanner used in capturing images from the sources. (ii) Internal resolution of the scanner. The internal resolution is the standard resolution used by the scanner to capture images from the sources. Images are extrapolated to different resolution from the ones captured using the internal resolution. This information, if available from the manufacturer, is also stored. (iii) The default file format in which the scanner stores the scanned image. (iv) Scanner setting like, gray level, color information, black & white page and bits per pixel information. (v) If the scanning is done using isolated pages or flattened pages (source is book). (vi) Copy right date of the source. (vii) Scanning center that digitized the sources. (viii) Digital publisher of the source (e.g. Digital Library Of India). (ix) Date on which the source is published digitally.

Annotation data: Figure 4 shows the hierarchy in which the annotation data is stored. The annotation information is stored at different levels like blocks, paragraphs, lines, words and Akshars. For each of these levels the image information is stored in separate fields. This separation between the image data and the annotation information eases updates or modifications to the schema.

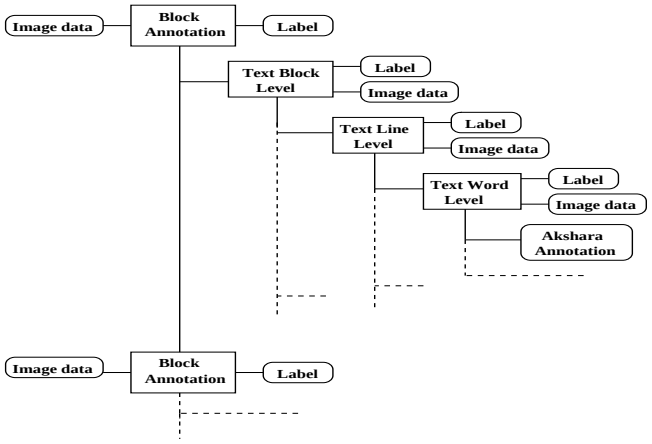

Fig. 4. Organizing the annotation data and image data in a hierarchy.

Page Structure Information: Segmentation of document image page is the primary step before annotation. In the first level of annotation i.e. block level annotation, the blocks are classified and labeled as explained below. Page structure information contains the bounding box information of every block, text and non-text obtained after segmentation, along with their labels after annotation. Storage of bounding box information instead of its image saves storage space and separates image data from the schema.

1. Text Blocks: Text blocks are labeled as paragraphs, heading, titles, captions, etc. The bounding box information is also stored in schema. Further text blocks are segmented into Lines(All lines in a block), words(all words in a line) and Aksharas(all aksharas present in a word). The bounding box information and annotation for each line and word is stored in the XML file.
2. Non-text Blocks: The non-text blocks that will be annotated and represented in the XML schema are :1)Mathematical equations.2)Figure/picture parts of the document image.3)Different types of graphs.4)Tables.

```
<line>
   <bbox>Bounding box of line</bbox>
   <word>Word Information </word>
    .
    .
    .
    .
    .
   <word>Word Information </word>
</line>
```

(a) Line

```
<bbox>
    <top> top coordinate</top>
    <left>left coordinate</left>
    <bottom>bottom coordinate</bottom>
    <right>coordinate</right>
</bbox>
```

(b) Bounding Boxes

```
<page>
    <cover>Name of meta information Page</cover>
    <pageNumber>Page number </pageNumber>
    <image>relative pathto page image in SS.</image>
    <text>Page text as seen in the page</text>
    <block>Information on a paragraph</block>
     .
     .
    <block>Information on a paragraph</block>
</page>
```

(c) Page

Fig. 5. XML Samples(a), (b) and (c)

A template of the annotation file for each page is given in figure 5(c). Each block consists of a number of lines of text i.e. a text paragraph. Each line information is stored as shown in figure 5(a). It consists of a group of information about each word present in it. The same schema is applicable for text blocks,words and sub-word components. Sub word components will have some more tags. The bounding box gives the coordinates of the rectangle containing the segments.Figure 5(b) gives the details of it.

Akshara Structure Information: Bounding box representation may lead to errors in representing Aksharas due to overlapping of the character or part of characters. Therefore, when akshara level information is required to be accessed from the schema only Aksharas have to be returned. Most classifiers require class identifiers (CIDs) of the components present in each akshara. The CIDs used by each group of researchers or users can be different from one another. Moreover number of unique components will vary from script to script. So a schema generation mechanism is the solution, where segmentation of sub-word components is done by each group with relevant informations. Then a schema generation code

will produce the annotation using the information. So to do a better storage, we will the following information is a must for a unique identification.

- Bounding boxes.
- Corresponding CID.
- Position in the word.

4 Software Architecture and Intermediate Process Specification

The OCR system is expected to be built in a modular fashion with each module being developed independently. We needed a mechanism so that each module could be seamlessly integrated to build the complete system. We devised an XML based scheme for unambiguous specification of intermediate data so that modules can be integrated in a generic framework of multi-lingual OCR. We illustrate this concept through simple abstracted representation of the OCR system.

The major architectural components of an Optical Character Recognition system would be: 1)Image acquisition,2)Image pre-processing/enhancement,3)Script-Independent processing and 4)Script-dependent schemes

XML has been used as an architecture specification language for each of these above modules and sub-modules. Each of the modules get a XML file as input and parsing it they extract the relevant information required for their functioning.

Image Acquisition: The input to an OCR is a raw image either scanned or taken from the corpus. The information related to the image that is needed by the OCR for choosing proper set of pre-processing routines are stored in the XML file. 1)Mode (single or multi page),2)Image Location,3)Image Name,4)Image format,5)Image dimensions,6)Scanning Resolution (200, 300 and 600 dpi),7)Pixel info type (color, grayscale and binary) and 8)Script identification is a list of properties stored. Figure 6(a) give the XML storage schema.

Pre-processing techniques: The input page may not be fit for applications involving character recognition. Hence we require enhancing the image quality for best results. Major pre-processing techniques used are noise removal, skew correction, Binarization. Corresponding to each technique we can have XML tags with their algorithm specific attributes. The input to these pre-processing routines is the XML file which contains all the image related information. The routines parse the XML file and pick the image from the location specified in the XML ImageURI tag. XML Schema for the preprocessing modules is given in Figure 6(b).

```
<xs:element name="RawImage">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="ImageName" type="xs:string"/>
   <xs:element name="ImageURI" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="mode" type="ModeType"/>
  <xs:attribute name="ImageEncodingFormat" type="xs:string"/>
  <xs:attribute name="ImageWidth" type="xs:positiveInteger"/>
  <xs:attribute name="ImageHeight" type="xs:positiveInteger"/>
  <xs:attribute name="ImagePixelInfo" type="PixelInfoType"/>
  <xs:attribute name="ScanningResolution" type="xs:positiveInteger"/>
  <xs:attribute name="NumberOfBytesPerPixel" type="xs:positiveInteger"/>
  <xs:attribute name="Script" type="xs"string" />
 </xs:complexType>
</xs:element>

<xs:simpleType name="ModType">
 <xs:restriction base="xs:string">
  <xs:enumeration value="singlepage"/>
  <xs:enumeration value="multipage"/>
 </xs:restriction>
</xs:simpleType>

<xs:simpleType name="PixelInfoType">
 <xs:restriction base="xs:string">
  <xs:enumeration value="GrayScale"/>
  <xs:enumeration value="Color"/>
  <xs:enumeration value="Binary"/>
 </xs:restriction>
</xs:simpleType>
```

(a) Raw Image Representation

```
<xs:element name="PreProcessedImage">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="RawImage">
    <xs:sequence>
     <xs:element name="Skew" type="SkewDetails"/>
     <xs:element name="Binarize" type="BinarizationDetails"/>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>

<xs:complexType name="SkewDetails">
 <xs:sequence>
  <xs:element name="ImageName" type="xs:string"/>
  <xs:element name="ImageURI" type="xs:string"/>
 </xs:sequence>
 <xs:attribute name="SkewAngleBefore" type="xs:decimal"/>
 <xs:attribute name="AlgorithmName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="BinarizationDetails">
 <xs:sequence>
  <xs:element name="ImageName" type="xs:string"/>
  <xs:element name="ImageURI" type="xs:string" />
 </xs:sequence>
 <xs:attribute name="ThresholdValue" type="xs:positiveInteger"/>
 <xs:attribute name="AlgorithmName" type="xs:string"/>
</xs:complexType>
```

(b) PreProcessing Techniques

Fig. 6. XML schema for Base Image representation and Preprocessing (a) and (b)

Script-Independent Processing: Script independent routine includes the following processes: 1)Text non-text separation 2)Text extraction 3)Text graphics separation

The above routines help in identifying picture/figure blocks, graphics and text blocks. It stores the bounding box information for each component in the XML tags (topLx, topLy, bottomRx and bottomRy). The routines also create tag related to type of component separated from the image like text, picture or graphics. Figure 7(a) and 7(b) shows the sample xml schema used.

The text blocks are further segmented into lines and words using script dependent techniques.

Script-dependent Processing: Making an OCR delivering high-performance output for Indian scripts would invariably involve making use of techniques tuned to script specific characteristics. Due to the widely varying nature of Indian scripts, such knowledge needs to be incorporated at the line segmentation state itself and throughout intermediate stages to the final output. Figure 5 shows the architecture for script dependent processing for Indian Scripts. This module will receive as input hypothesized line and word boundaries from script independent component. These hypotheses will be refined/modified by the script dependent processing. The different modules in this architecture are:

1. **Line Segmentation**: The presence of ascenders and descenders in some Indian scripts make it difficult to demarcate the text lines using a simple

```
<xs:element name="PictureBlock" type="SegmentedBlock"/>
<xs:element name="GraphicsBlock" type="SegmentedBlock"/>
<xs:element name="TextBlock" type="SegmentedBlock"/>
<xs:element name="SegmentedBlock">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="BLOCK"/>
   <xs:element name="InputImageURI" type="xs:string" />
   <xs:element name="OutputImageURI" type="xs:string" />
  </xs:sequence>
  <xs:attribute name="TotalNumber" type="xs:positiveInteger"/>
 </xs:complexType>
</xs:element>
```

(a) Script Independent Block

```
<xs:element name="BLOCK">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Unicode"/>
   <xs:element ref="TextLine"/>
   <xs:element name="RejectClass" type="xs:string"/>
   <xs:element name="topLx" type="xs:positiveInteger"/>
   <xs:element name="topLy" type="xs:positiveInteger"/>
   <xs:element name="bottomRx" type="xs:positiveInteger"/>
   <xs:element name="bottomRy" type="xs:positiveInteger"/>
  </xs:sequence>
  <xs:attribute name="Number" type="xs:positiveInteger"/>
  <xs:attiribute name="SemanticLabel" type="xs:string"/>
 </xs:complexType>
</xs:element>
```

(b) BLOCK

```
<xs:element name="Unicode">
 <xs:complexType>
  <xs:attribute name="FileURI" type="xs:string"/>
  <xs:attribute name="FontSize" type="xs:positiveInteger"/>
 </xs:complexType>
</xs:element>
```

(c) Unicode Text

```
<xs:element name="TextLine">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="BLOCK"/>
   <xs:element ref="TextWord"/>
  </xs:sequence>
  <xs:attribute name="TotalNumber" type="xs:positiveInteger"/>
 </xs:complexType>
</xs:element>
```

(d) TextLine

Fig. 7. XML Schema for Segmentation and Recognition Modules(a), (b), (c) and (d)

cue like interline white space separation. A technique robust enough to deal with the merging of text lines at the ascenders or descenders is required. These techniques return the line boundaries which are stored in the XML.

2. **Word Segmentation**: Word segmentation can be reliably done using the white space separating adjacent words. These again return the word boundaries.
3. Each word is further subjected to symbol extraction using script specific information like character conjuncts, matras, half characters, ligatures, etc. Next each symbol goes through classification process where certain features for each isolated character is computed and each character is classified as either true class (correct recognition), wrong class (substitution error) or an unknown class(rejection error).

Thus the tags that are created by the script dependent process for representing its data are: 1)Line boundaries 2)Word boundaries 3)Unicode for the text region 4)Markers for Reject class XML template for the script dependent processing is shown in Figures 7(d),7(c) and 7(b). Template for *TextWord* is same as *TextLine*.

The final Unicode is associated with the with a word image and not akshara. The text word can ve classified as Reject- class if it cannot be reliably recognized as a valid word. Storing all intermediate information and final results in XML helps in saving memory as well as for verifying the performance of recognition algorithms.

The examples illustrated above are indicative. For each of the functional module similar XMl based I/O specification have been designed.

5 Multi-Script data Handling

In order to handle books/document images with multiple scripts we create a script specific tag in the XML file generated during annotation or early stages of OCR. A trainable Gabor filter based script identification or script identification based on edge based features helps in indentifying scripts for annotating textual blocks with a script specific information in a multilingual collection of documents. XML tags generated are shown in Figure 6(a).

Knowing the scripts present in a document image in the early stages of document image we can automate the application of corresponding script OCR after document has undergone pre-processing and text extraction processes. In case of a document image with multiple scripts, two situations can occur. We can identify the script for each textual block after segmentation. However, this approach does not eliminate the problem when we have multi-script line. In this case, words tagged with reject tag of a classifier are further analysed to identify script and accordingly we replace the reject tag with script specific identifier.

6 OCRed Output Specification

The final output received from the OCR system is encoded using XML tags. The output XML file has the document layout structure, segmented regions of interest and their categorization (text, graphics, pictures) and the UNICODE stream for the words for the text extracted. The information content of the final XML output of the document-analysis-cum-OCR module in figure 8. There are 4 levels of granularity of descriptions for the document image.

1. The level-0 corresponds to the complete document image.
2. Level-1 corresponds to next granular level arising after the segmentation. The entities in the level-1 are text regions, picture regions, and graphics or tables.
3. The level-2 granularity exists only for the text regions. It comprises the segmented lines of text.
4. At level-3 we have the words. And finally the OCR output unicode stream of symbols for the corresponding word.

For each level the data representation format using XML has been defined in the section above.

Figure 6. 4 level of granularity describing the final XML output of the document-analysis-cum-OCR module

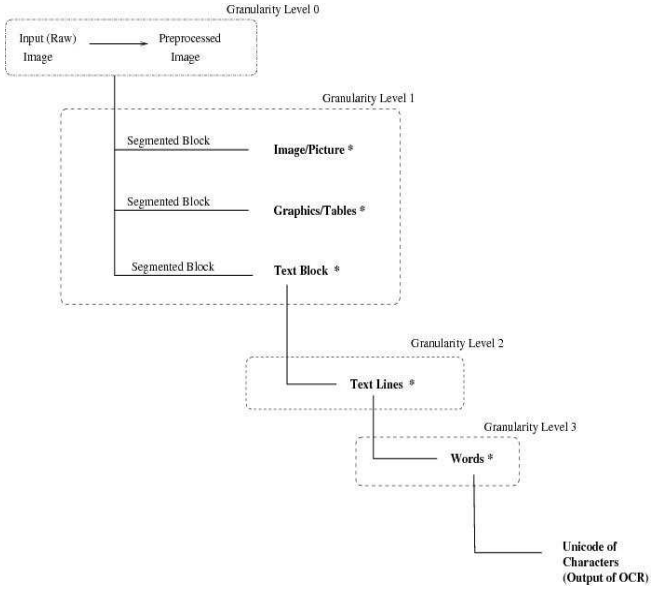

Fig. 8. 4 level of granularity describing the final XML output of the document-analysis-cum-OCR module.

6.1 Presentation Engine

Presentation engine can be used as an interface to display the electronic version of the paper document which has been OCRed. This interface is also capable of providing options like rendering, editing and conversion of of the electronic document to other formats (rtf, doc, etc.). Typical features of word-processors can be used for editing purpose.

The electronic document can be rendered in a web-browser. But the XML output cannot be directly used by the existing browsers for display. Hence it needs to be transformed into a form compatible for browse r display. The content of an XML element, such as an abstract or a paragraph, has no explicit style or format. Since XML is not concerned with visualization aspects, it is necessary to specify the element rendering in a different language. XSL (XML Style-Sheet Language) is a language used for expressing style sheets. An XSL style sheet specifies the presentation of a class of XML documents by specifying how an instance of the class is transformed into an XML document that uses the formatting vocabulary. The reference to an external style sheet is reported in the first row of the XML file.

```
< ? XML-stylesheet href=tami1.XSL type=text/xsl ? >
```

An XML style sheet processor accepts a document in XML and XSL style sheet and produces the presentation of the XML source content that was intended by the designer of that style sheet. Following the W3C recommendations, the selection of font, font-size, and alignment is not specified in the XSL file, but in the Cascading Style Sheet (.css) file which is unique for each class of documents. The fonts need to be selected appropriately to ensure that the relative font sizes of various text portions in the image is preserved and also fits within the area assigned to a textual entity.

Also The transformed document in HTML/XML formulate aggregates all information about textual, graphical, layout, and other logical/structural information extracted from document analysis process. Since the XML-output from the OCR module specifies locations of various physical entities in a document page in terms of the bounding box coordinates, this kind of representation cannot be used by the browser since neither HTML nor XML allow coordinate-based positioning of text and images on the screen. The page layout can be rendered by means nested tables. This is done by recursively judging the best possible approximation of layout of sub-entities within an entity in terms of columns or rows. Nested tables are formed to correspond to the given layout. After mapping the entity locations to hierarchical tabular layout, the next step is to choose a suitable font (size and style) for displaying the textual entities. The entities having image (picture/graphics) are drawn on the corresponding regions using image data read from the URI location (a .jpg or .gif file). HTML allows the font information to be specified in the HTML document itself. In fact the font information can be specified for every text portion. A disadvantage of HTML is that the logical structure of the document page cannot be represented. HTML allows only a fixed set of tags. Hence all other tags pertaining to information other than physical layout must be removed from the OCR-output XML.

7 Ontology Linkage and Annotation Management

With a large corpus of multilingual data, along with multilingual summarization and translation tools, a well-directed research effort would be needed to ensure concept- and content-based retrieval of content from across multilingual documents. The document image representation in XML can facilitate linkage of different image components with ontology for conceptual querying, browsing and hyper-linking across languages and scripts. Documents image ontology establishes the semantics of logical components of the document image and the relationships amongst them. Also, onotlogy specific semantics of meta-data associated with documents. We can use document class specific ontologies to generate automated hyper-linking between relevant documents and document components. For instance using the semantics of the concept “author” we can hyperlink pieces originating from the work of the same author(his own writing as well as translations of his work in other languages)

We can represent concepts in term of document image based features, OCRed output and attributes/ keywords (derived from metadata). Many conceptual entities manifest in document images in the form of observables (media observables), for example word image features, keywords, words in different scripts (recognized by the language specific OCR) etc.Document ontology help in associating media observables to concepts and allow propagation of media properties from concept to another. In order to meet requirements of multimedia data we use an extension of OWL called multimedia OWL(MOWL) [15]. In MOWL we represent concepts in terms of observation model (OM) which relates concepts to observables in the media domain. Figure 9 depicts a possible observation model for a steam locomotive. In general, an OM for a concept comprises not only the media properties of the concept but also the media properties of related concepts, e.g. pair of lines signifying railway tracks in this example. Moreover, a concept can have many alternative manifestations in a media form, e.g. many different possible body shapes of a steam locomotive. OM is in fact a belief network with conditional probabilities attached to the links.

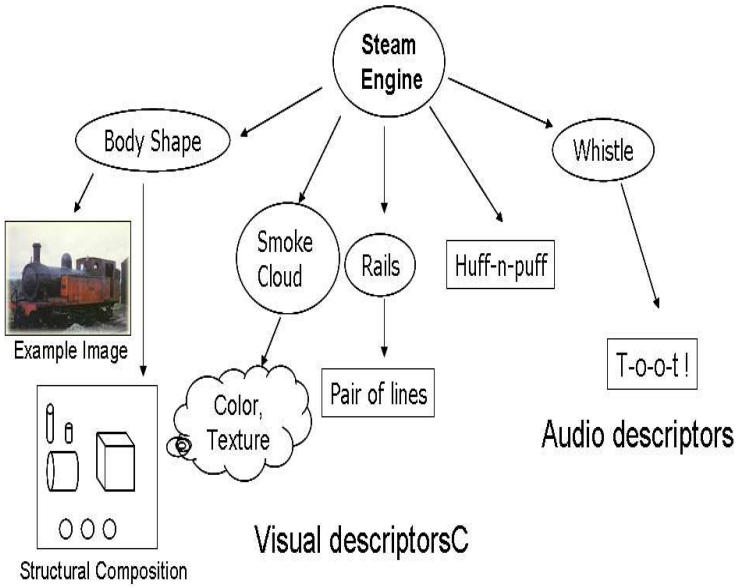

Fig. 9. Concept specification in MOWL.

MOWL uses probabilistic reasoning for dealing with uncertainties in the multimedia domain. This feature of uncertainty handling makes M-OWL suitable for dealing with OCRed text which may be noisy due to errors in recognition. Using M-OWL ontology we can recognize concepts through integration of probabilistic

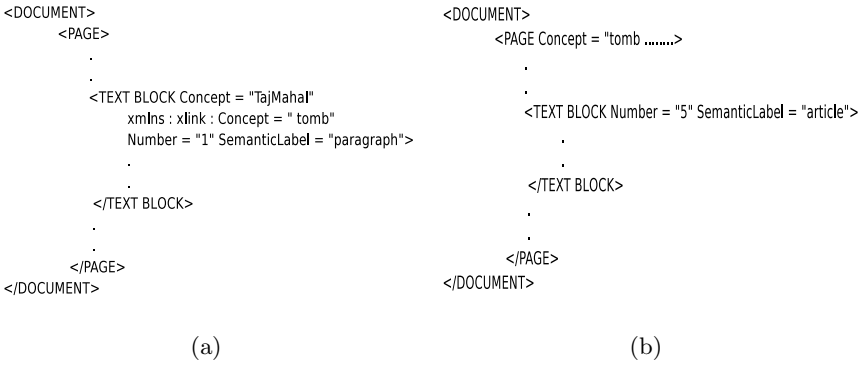

Fig. 10. XML tagging structure based MOWL/OWL representation

evidences obtained from noisy keywords detected in the text, meta-data about the document, pictures and graphics present in the document image. Since, leaf nodes of observables can be keywords from different languages, using MOWL ontology we can spot same or similar concepts in images of documents across scripts. We explain the process of concept spotting with the help of an example. Let us consider a simple concept Tomb in the MOWL ontology. "TajMahal" is another concept associated with the node Tomb through IS_A link., Each concept node has observation model associated with it. In fact Tajmahal inherits OM of Tomb. In the observable set of Tajmahal we can have keywords like Taj, Tajmahal, Agra, etc. in different scripts. In case of Tomb observable set can contain synonyms of Tombs in different languages. Also, images with relevant captions and image/graphics based features can also be part of observables. We look for these observables in the OCRed document, integrate and propagate evidences for recognition of concepts on the basis posteriori belief accumulated. Concept spotting involves application of concept recognition algorithm at different levels of granularity of the document starting from paragraph level. We consider, paragraph, page, chapter and book level in a structural hierarchy. Conceptual annotations thus obtained using M-OWL ontology is added to the documents using additional XML tags. Tags associated at different levels of document hierarchy. Also tags reflect the conceptual hierarchy, if any, detected during concept spotting. For example, a page conceptually tagged with the label Tajmahal also gets the tag Tomb with the hierarchical relation indicated. The tagging structure is consistent with XML based MOWL/OWL representation (Figure 10). XML based representation of document images has facilitated ontology driven conceptual tagging of OCRed text in the similar way as semantic web enablement is happening for text, image, video, etc. These conceptual tags will enable us to build conceptual search engine that can work across scripts. These tags are also used for hyperlinking. URI of the hyperlinked document is also made part of the XML representation of the document.

8 Conclusions

In this paper, we have presented a scheme for management of multilingual OCR system using XML as the universal medium for data description and exchange. We have shown that our XML based representation can handle different requirements ofdf multilingual document image managementsystems. Using XML based unified representation framework we have provided soluutions for groundtruth management, moduler multiparty OCR developement and semantics or concept based search of OCR'ed documents. Out approach provides a generic solution which can be emulated for other multilingual multiscript OCR envornments.

References

1. Breuel, T., Kaiserslautern, U.: The hocr microformat for ocr workflow and results. In: Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on. Volume 2. (Sept. 2007) 1063–1067
2. International Unipen foundation: The unipen project. http://www.unipen.org (1994)
3. Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., Janet, S.: Unipen project of on-line data exchange and recognizer benchmarks. In: Pattern Recognition, 1994. Vol. 2 - Conference B: Computer Vision and Image Processing., Proceedings of the 12th IAPR International. Conference on. Volume 2. (Oct 1994) 29–33 vol.2
4. Houlding, S.W.: Xml – an opportunity for ¡meaningful¿ data standards in the geosciences. Computers & Geosciences **27**(7) (2001) 839 – 849
5. Agrawal, M., Bali, K., Madhvanath, S., Vuurpijl, L.: Upx: a new xml representation for annotated datasets of online handwriting data. In: Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on. (Aug.-1 Sept. 2005) 1161–1165 Vol. 2
6. A. Bhaskarbhatla, S. Madhavanath, M. Pavan Kumar, A. Balasubramanian and C. V. Jawahar: Representation and Annotation of Online Handwritten Data. In: Proc. of 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR). (2004) 136–141
7. W3C Mullti-modal Interaction Working Group: Ink markup language (inkml). http://www.w3.org/2002/mmi/ink (2003)
8. C. V. Jawahar and Anand Kumar: Content Level Annotation of Large Collection of Printed Document Images. In: Proc. of International Conference on Document Analysis and Recognition (ICDAR). (2007) 799–803
9. C. V. Jawahar, Anand Kumar, A. Phaneendra, K. J. Jinesh: Building Data Sets for Indian Language OCR Research. Springer Series in Advances in Pattern Recognition (2009)
10. Wrede, S., Fritsch, J., Bauckhage, C., Sagerer, G.: An xml based framework for cognitive vision architectures. In: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Volume 1. (Aug. 2004) 757–760 Vol.1
11. W3C Web Ontology Working Group: Web Ontological Language (OWL). http://www.w3.org/TR/owl-guide/ (2004)
12. Mallik, A., Pasumarthi, P., Chaudhury, S.: Multimedia ontology learning for automatic annotation and video browsing. In: MIR '08: Proceeding of the 1st ACM

international conference on Multimedia information retrieval, New York, NY, USA, ACM (2008) 387–394
13. Yoon, J., Kim, S.: Schema extraction for multimedia xml document retrieval. In: Web Information Systems Engineering, 2000. Proceedings of the First International Conference on. Volume 2. (2000) 113–120 vol.2
14. Lear, A.: Xml seen as integral to application integration. IT Professional **1**(5) (Sep/Oct 1999) 12–16
15. Ghosh, H., Harit, G., Chaudhury, S.: Ontology based interaction with multimedia collections. ICDL'06, International Conference on Digital Library (2006) –