Institute of Computing/*Instituto de Computação*
University of Campinas/*Universidade Estadual de Campinas*

Shadows: a new means of representing documents

Matheus Silva Mota

This text corresponds to the final version of the Dissertation duly corrected and defended by Matheus Silva Mota and approved by the Board of Examiners.
Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Matheus Silva Mota e aprovada pela Banca Examinadora.

Campinas, June 24, 2012.

[illegible]
Prof.ª Dr.ª Claudia Maria Bauzer Medeiros
Institute of Computing - UNICAMP
(Supervisor/*Orientadora*)

Dissertation presented to the Institute of Computing of UNICAMP in partial fulfillment of the requirements to be awarded the MSc of Computer Science.
Dissertação apresentada ao Instituto de Computação, UNICAMP*, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.*

FICHA CATALOGRÁFICA ELABORADA POR
ANA REGINA MACHADO - CRB8/5467
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Mota, Matheus Silva, 1986-
M856s Shadows : a new means of representing documents / Matheus Silva Mota. – Campinas, SP : [s.n.], 2012.

Orientador: Claudia Maria Bauzer Medeiros.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Banco de dados - Gerência. 2. Sistemas de recuperação da informação - Documentos. I. Medeiros, Claudia Maria Bauzer, 1954-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: Shadows : a new means of representing documents
Palavras-chave em inglês:
Database management
Information storage and retrieval systems - Documents
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Claudia Maria Bauzer Medeiros [Orientador]
André Santanchè
Angelo Roncalli Alencar Brayner
Data de defesa: 18-05-2012
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 18 de Maio de 2012, pela Banca examinadora composta pelos Professores Doutores:

[illegible]

Prof. Dr. Angelo Roncalli Alencar Brayner
UNIFOR

André Santanchè

Prof. Dr. André Santanchè
IC / UNICAMP

[illegible]

Prof^a. Dr^a. Cláudia Maria Bauzer Medeiros
IC / UNICAMP

Institute of Computing/*Instituto de Computação*
University of Campinas/*Universidade Estadual de Campinas*

Shadows: a new means of representing documents

Matheus Silva Mota[1]

May 2012

[1]Financial support: CNPq scholarship (process 133815/2010-2) 2010–2012

Abstract

Document production tools are present everywhere, resulting in an exponential growth of increasingly complex, distributed and heterogeneous documents. This hampers document exchange, as well as their annotation and retrieval. While information retrieval mechanisms concentrate on textual features (corpus analysis), annotation approaches either target specific formats or require that a document follows interoperable standards – defined via schemas. This work presents our effort to handle these problems, providing a more flexible solution. Rather than trying to modify or convert the document itself, or to target only textual characteristics, the strategy described in this work is based on an intermediate descriptor – the *document shadow*. A shadow represents domain-relevant aspects and elements of both structure and content of a given document. Shadows are not restricted to the description of textual features, but also concern other elements, such as multimedia artifacts. Furthermore, shadows can be stored in a database, thereby supporting queries on document structure and content, regardless document formats.

Resumo

Ferramentas de produção de documentos estão cada vez mais acessíveis e sofisticadas, resultando em um crescimento exponencial de documentos cada vez mais complexos, distribuídos e heterogêneos. Isto dificulta os processos de troca, anotação e recuperação de documentos. Enquanto mecanismos de recuperação da informação concentram-se apenas no processamento de características textuais (análise de *corpus*), estratégias de anotação de documentos procuram concentrar-se em formatos específicos ou exigem que o documento a ser anotado siga padrões de interoperabilidade – definidos por esquemas. Este trabalho apresenta o nosso esforço para lidar com estes problemas, propondo uma solução mais flexível para estes e outros processos. Ao invés de tentar modificar ou converter um documento, ou concentrar-se apenas nas características textuais deste, a estratégia descrita nesta dissertação propõe a elaboração de um descritor intermediário - denominado *shadow* – que representa e sumariza aspectos e elementos da estrutura e do conteúdo de um documento que sejam relevantes a um dado domínio. *Shadows* não se restringem à descrição de características textuais de um documento, preservando, por exemplo, a hierarquia entre os elementos e descrevendo outros tipos de artefatos, como artefatos multimídia. Além disto, *Shadows* podem ser anotados e armazenados em bancos de dados, permitindo consultas sobre a estrutura e conteúdo de documentos, independentemente de formatos.

Acknowledgements

First and foremost, I would like to thank Professor Claudia Medeiros, for the example of dedication, for all opportunities, for all advices that made me grow professionally and personally, for all scolding and caring, for believing in my research potential and for all time and patience dispensed to me in order to introduce me to the art of scientific research.

I would like to thank the amazing members of the Laboratory of Information Systems (LIS), for the moments of learning, for the patience, for sharing moments of difficulty, for the good laughs and for the friendship consolidated over the years.

I would like to thank colleagues, faculty and staff of the Institute of Computing and of UNICAMP, for all the attention, for the moments of learning and for the companionship that created a healthy environment for the development of this research.

I would like to thank my parents, Olga and Antônio, and my sister, Laís, for the infinite love, for all support and dedication, for being my safe harbor in times of storm, for always putting a smile on my face even when facing many difficulties, for the example of faith and for always believing in my potential.

I would like to thank my good friend Ivo Koga, for sharing moments of difficulty and joy, for all laughs and for all attention that were fundamental throughout this period.

I would like to thank my huge and amazing family, for all support and love.

I would like to thank my good friend Ricardo, for his presence, even at distance, for the support, for sharing moments of difficulty and joy, for all laughs and for the friendship.

I would like to thank Jessica, Augusto, Neide, Rita, Thainná, Dona Graça, Nícolas e Diego, for constant support and for all important and unforgettable attitudes and words.

I would like to thank my housemates, for their patience and attention, for the good laughs and for welcoming a new comer to Campinas.

I would like to thank CNPq[2], CAPES, FAPESP and the INCT in Web Science for the financial support.

I would like to thank many other people that were not mentioned but also believed, supported, participated, collaborated with this work and nevertheless remain anonymous in these acknowledgments.

[2]For the scholarship – process 133815/2010-2

Contents

List of Figures

Chapter 1

Introduction and Motivation

The Web has become a huge platform for document publishing, with easy access to sophisticated document production tools. The evolution and multiplication of these authoring tools brought about the proliferation of document formats, resulting in an exponential growth of increasingly complex, distributed and heterogeneous documents.

Ideally, document production tools should produce interoperable documents. However, in most cases, such tools have not been conceived to produce files with explicit structure. They strongly couple the content to the file structure and software representation [44, 24, 43]. Furthermore, document production tools have increasingly been offering support for more than flat text, handling also artifacts such as charts, tables or multimedia elements. This further increases the problem of document heterogeneity and complexity.

In a scenario with high diversity of non-interoperable formats and a large volume of complex documents, challenges arise when it comes to management, storage and retrieval techniques, correlation algorithms and new methodologies to present, annotate and mine documents and their content. In addition, there are problems related to documents produced to be used in multiple contexts – for instance, in the context of scientific research, participating research groups have different needs of document handling [24].

Document management and retrieval systems use three main strategies to deal with large volumes of complex and heterogeneous documents [27, 29, 9]. The first strategy supports only some specific file format, making it necessary to convert the original document to the supported format. The second strategy requires documents that follow interoperable standards (e.g., XML) or structures. The third strategy considers a document to be a general digital artifact, supporting only metadata and requiring user assistance. The first strategy presents problems when original file preservation is needed. In strategy two, the main difficulty is to handle format diversity, since interoperable formats and predefined schemes are a prerequisite. On the other hand, approach three deals very well with the diversity of file formats, but provides limited support to indexation, retrieval and annotation.

This dissertation presents *Shadow-driven Representation* (SdR), a novel strategy to represent documents independently of format, preserving the original file and handling large volume of documents. A *shadow* is an interoperable document descriptor that summarizes key aspects and elements of a document, preserving their structural relationships. These elements (e.g., sections, tables, embedded multimedia artifacts, references) are defined by users (e.g., research groups may have different interests), and thus one document may have many shadows. Once a set of elements of interest is defined, shadows are instantiated based in this set. Unlike other approaches in the literature that restrict document description to text, shadows consider other kinds of elements within a document, such as tables or images, thereby supporting a wide variety of operations and correlations. Though we have implemented shadows as XML documents stored in a database (our *shadow base*), this is just a possible materialization of the concept, which transfers document querying tasks to the DBMS.

The main advantages of the SdR approach are: (i) a *shadow* isolates domain-relevant elements in a document, regardless of formats; (ii) *shadows* can have have different granularity levels and concern distinct types of elements, depending on the domain needs; and (iii) *shadows* follows established interoperability standards, allowing automatic and semi-automatic machine consumption.

This dissertation also presents a validation of the SdR strategy, discussing problems and solutions of a full SdR implementation. In addition, this work presents a case study, where shadows are used to produce semantic annotations that link documents concerning biodiversity studies to open data on the Web. This dissertation was developed in the *LIS - Laboratory of Information Systems* - Institute of computing of UNICAMP.

The main contributions of this dissertation are:

- Proposal and specification of a technique for document representation that supports large volumes of heterogeneous documents;

- A validation of the proposal via an implemented prototype, that covers the full cycle of shadow specification, creation, annotation and management;

This research led to the following publications:

- "*Shadow-driven Document Representation: A summarization-based strategy to represent non-interoperable documents*". Matheus Silva Mota and Claudia Bauzer Medeiros. XI Workshop on Ongoing Thesis and Dissertations WebMedia, 2011.

- "*Using linked data to extract geo-knowledge*". Matheus Silva Mota, João Sávio C. Longo, Daniel Cintra Cugler and Claudia Bauzer Medeiros. XII Brazilian Symposium on GeoInformatics (GeoInfo), 2011. (Received the Best paper Award) [35]

The text is organized as follows. Chapter 2 introduces concepts and related work. Chapter 3 presents a detailed explanation of the SdR approach. Chapter 4 presents our implementation of the SdR strategy and discusses implementation details. Chapter 5 presents a case study where we use shadows to allow semantic annotations of documents in the biodiversity context. Finally, Chapter 6 presents conclusions and ongoing work.

Chapter 2

Basic Concepts and Related Work

This chapter presents basic concepts related to this dissertation, and related work. Section 2.1 describes resource descriptors, such as image descriptors (Section 2.1.1) and metadata (Section 2.1.2). Section 2.2 discusses research on document management, while Section 2.3 concentrates on document annotation and retrieval. Section 2.4 presents concepts and technologies related to the Semantic Web, such as semantic annotations (Section 2.4.1) and linked data and entity linking (Section 2.4.2). Finally, Section 2.5 presents conclusions.

2.1 Resource Descriptors

2.1.1 Image descriptors

An Image Descriptor is a data structure that summarizes the content of an image. According to [10], an image descriptor can be defined as a pair composed of a feature vector and a distance function. The feature vector represents a set of properties (e.g, shape, color, texture) extracted from images. The distance function (or similarity) is used to compare feature vectors through a specific metric [32, 30]. Figure 2.1 shows the main components of an image descriptor, and how they are used to compare two images. Here, the descriptor is a pair $< \epsilon_D, \delta_D >$.

To extract visual properties, image processing algorithms usually focus on specific characteristics of an image and mainly follow two steps: (i) points of interest are identified and pass through a feature extraction process; and (ii) values are computed based on each point of interest, according to the type of information that needs to be extracted or recognized [30, 1].

Image descriptors have two main advantages: (i) the features extracted can be stored for subsequent processing; and (ii) different image descriptors (e.g., based on color, shape,

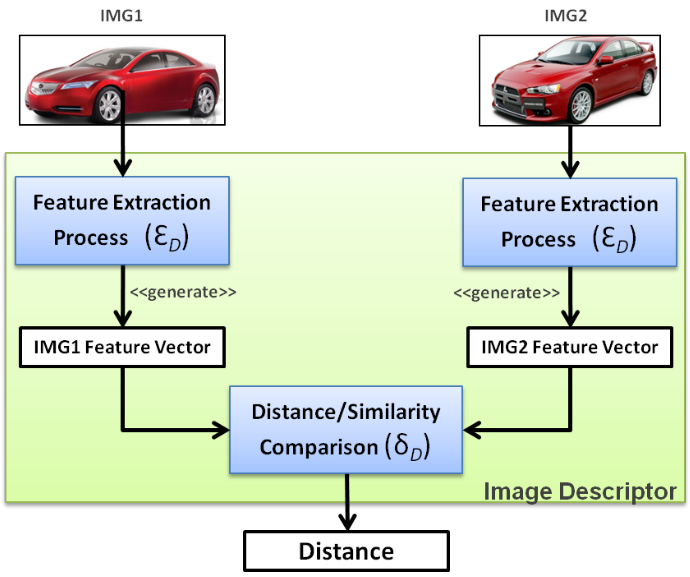

IMG1
IMG2
Feature Extraction Process ($\mathcal{E}_D$)
Feature Extraction Process ($\mathcal{E}_D$)
<<generate>>
<<generate>>
IMG1 Feature Vector
IMG2 Feature Vector
Distance/Similarity Comparison (δ_D)
Image Descriptor
Distance

Figure 2.1: Components of an image descriptor (according to [10])

texture and others) can be combined, implying on scalability [32, 15, 34]. These advantages can be clearly noted on applications that process large volumes of images (mainly indexing and retrieval). Basically, these applications pre-process each image and generate/store feature vector(s), according to the application needs. Later, instead of performing management tasks over the image itself – which can be costly –, applications process its feature vector.

Image descriptors are particularly helpful in understanding our shadow-driven representation (SdR) approach. As will be seen in Chapter 3, rather than looking for matches of metadata or annotations, or opening a document to extract specific characteristics – which is the usual approach in document management systems –, the SdR strategy pre-processes and extracts points of interest (key elements) of a document. Then, based on the extracted features, we generate a structure that describes the document. Like image descriptors, *shadows* are stored apart of the documents themselves, and can be used to process them – Section 5 presents case studies where we use shadows to indirectly annotate and retrieve documents, independently of file formats.

2.1.2 Metadata and Metadata Standards

Metadata can be seen as a high level description of data, providing an organization of descriptions of digital or non-digital resources [23]. In this section, we discuss metadata focusing only on metadata for digital objects, in information systems.

Metadata, or meta-information, is a structured information and regulatory tool to explain, locate, identify and describe resources, allowing information exchange/integration and helping users or management tools [50]. Metadata are usually associated with retrieval tasks.

These "data about data" or "information about information" provide semantics and can be associated with some resource or parts thereof [18, 13]. In the literature, metadata are mainly related to the following purposes: resource description; information retrieval; information exchange (interoperability); management of information (lineage, trust); rights, ownership and authenticity management [23]. Metadata information are classified according to their function (descriptive, structural, administrative, rights management, preservation) and to its level of semantic abstraction (low-level and high-level) [38].

Low-level metadata (usually more technical, e.g., file-type, an image width or a file size) have less value for end users. However, they are commonly used by information systems in order to support simple management tasks (since data types are usually primitive, e.g., integers, floats and predefined strings). **High-level metadata** – semantically rich descriptive information – are more interesting to end users, since they can describe semantic entities (events, concepts, states, places). Nevertheless, such information is usually provided by humans as free text or tags, hampering automatic machine consumption.

Both high and low-level metadata fields are not designed to support sophisticated management tasks, but are widely used as input to several approaches that process this information in order to produce indices, descriptions, clusters etc. In information systems, the appearance of metadata can be divided in the following levels [23].

- **Physical level:** stored/stream of bits and bytes. This level is usually hidden from applications and users. At this level, information systems focus on optimizing record allocations in file systems, compression and other raw-data tasks;

- **Logical level:** database management systems level. Here, a more technical metadata schema (which can be defined via some information model, e.g., Relational Data Model) can be written via some data definition language. Metadata instantiation should follow these definitions;

- **Programming/representation:** the metadata schema expressed in code (of some programming language), transforming metadata instances in instances of the application domain. At this level, metadata can be persisted via some mark-up languages (XML) or other data representation tool;

- **Conceptual:** domain entities with their attributes and relationships. At this level, Entity-Relationship model or Unified Modeling Language can be adopted to represent real-world entities and their attributes and relations.

According to [23], and represented in Figure 2.2, there are three metadata building blocks:

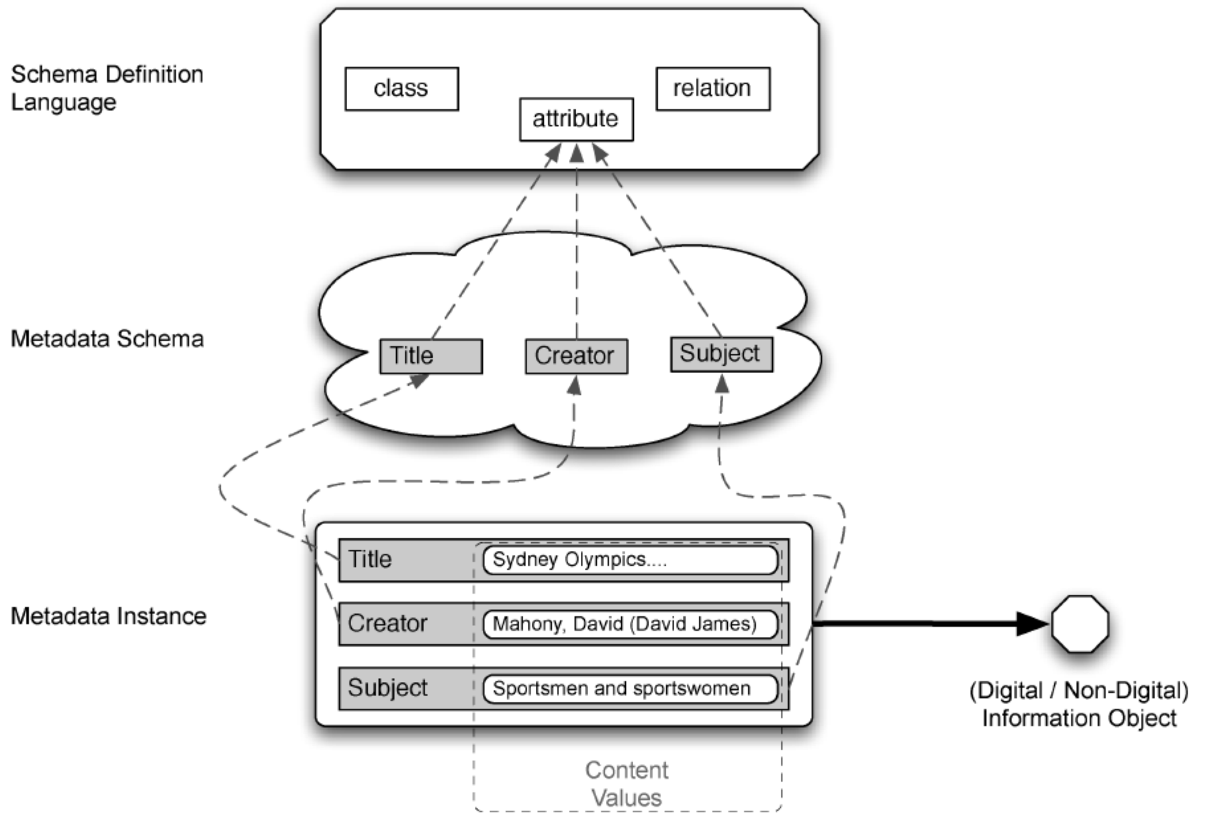


Figure 2.2: Overview of the three metadata building blocks (source [23])

1. **Schema Definition Language:** the domain specific metadata schema must be represented in some schema definition language (e.g., XML Schema, SQL-DDL, RDFS, OWL, UML).
2. **Metadata Schema**: set of metadata elements – or fields – and their names and (optional) encoding/syntax rules.
3. **Metadata Instance:** the information about the digital object itself, associated with a predefined element of the Metadata Schema.

Different domains and needs may require distinct metadata vocabularies. Metadata standards propose and define a set of elements that improves data sharing and integration among different users and applications.

As presented in Chapter 3, a shadow is a document descriptor that should contain domain relevant elements extracted from documents [50, 23]. Our solution to create and persist this descriptor produces a hierarchical set of metadata instances (following widely adopted metadata standards), according to the set of elements of interest defined by a user-produced schema. As presented in Chapter 4, we use a set of metadata standard initiatives related to documents and other kinds of digital objects – that usually appear inside documents – to define types of elements within documents in a shadow.

2.1.3 Other Descriptors

Other research areas also adopt the notion of *descriptors* in order to perform computational tasks. Mainly, those areas use descriptors for more efficient processing and queries – content-based retrieval, for instance. Some of those descriptors are:

DNA and protein sequences descriptor: Since they are large sequences, many researchers propose the notion of descriptors (also known by domain experts as suffix vectors) for both DNA and protein sequences. Such work focus, for instance, in performing more efficient queries on biological databases [37] or allowing more efficient similarity discovery [41]. Basically, this kind of descriptor summarizes the sequences preserving representative structural features that characterize a species. Later, instead of processing the sequence itself, those descriptors are processed.

Descriptors for video file processing: There are two main approaches [25] to produce a summary or execute a content-based query for video files: (i) Producing an image (usually very large, depends on the input file and on the strategy adopted) that represents the whole video and its temporal events [47, 49]; and (ii) selecting a key frame for the whole video or a key frame [45, 7, 42] for specific parts of the video – there are different ways to split the video. In both approaches, an image is produced or selected in order to allow more efficient queries (frame by frame comparison is costly). Instead of processing the video itself, those systems process the generated images, also using image descriptor. Therefore, such images act like a video descriptor.

Figure 2.3, for instance, is an example of image that represents the *visual rhythm* of a video. Basically, the visual rhythm is a combination of slices of frame sequences (Figure 2.3 specifically represents a sequence of captioned frames in a video).

2.2 Document Management

As stressed in Chapter 1, document management and retrieval systems use three main strategies to deal with large volumes of complex and heterogeneous documents [27, 29, 9]. The first strategy supports only some specific file format, making it necessary to

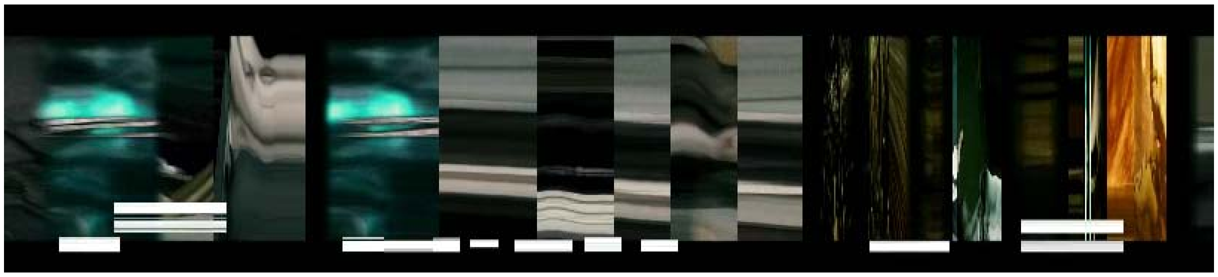


Figure 2.3: Example of a generated image that represents the visual rhythm of a video (source [49])

convert the original document to the supported format. The second strategy requires documents that follow interoperable standards (e.g., XML) or pre-defined schemas. The third strategy considers a document to be a general digital artifact, supporting only metadata management and requiring user assistance. The first strategy presents problems when original file preservation is needed. In strategy two, the main difficulty is to handle format diversity, since interoperable formats and predefined schemes are a prerequisite. Approach three deals well with file format diversity, but provides limited support to indexation, retrieval and annotation.

Document retrieval systems widely adopt automatically generated semantic annotations or free manual annotations to support indexation and retrieval [27, 29, 9]. Existing tools for manual document annotation can be divided in two categories. The first one produces a file (stored in or with the document) that contains annotations and/or modifies the original document by inserting annotations into it. This approach hampers the process of indexation, exchange and sharing of annotations. The second category is Web-based, holding documents and annotations on a database or library. Once the annotations are produced, they can be shared with a specific group of users or can be used by other applications.

Independently of where annotation are made, annotation tools allow three main annotation strategies. The first strategy concentrates on some specific file format, converting the original document and annotating the converted file. The second strategy requires a file that follows some interoperable standard (e.g., XML). On the other hand, the third strategy deals with documents as images, allowing annotations by floating free-shape layers. Those approaches present problems when there is need for, respectively (i) original file preservation; (ii) format diversity handling; and (iii) document specificity and internal artifacts handling As will be seen, shadows are a means to solve these issues.

2.3 Document Annotation and Retrieval

Shadows describe documents, and thus must be compared with document description and extraction techniques found in the Information Retrieval (IR) and database literature. IR is primarily concerned with textual evidence. There are countless techniques to extract relevant keywords, concepts, sentences from a document in order to represent or describe it. There is also the need for a corpus that defines the basis for extraction algorithms.

Representation structures may be used to index, rank and retrieve documents. Once these structures are created, sets of documents can be clustered according to them – e.g., to correlate documents, or summarize them. These structures can also be used to identify documents that are representative of a set – a set of documents can be summarized by a document, i.e., by the structure that represents a document, e.g., [21].

Another means to represent documents in IR is the use of metadata, often taking advantage of metadata standards like Dublin Core [11]. In particular, if documents are written in XML, then element tags can also be used (and in this case they are sometimes called *facets* [58]). The same kind of strategy is adopted by [54], that represents a document by a vector of concepts, and then tries to reduce the dimensionality of this vector to speed up document retrieval, clustering and comparison.

While most IR research considers documents immutable, recent efforts are being undertaken to update the representation structures when documents are updated (e.g.,[53]).

Web database work on document management is intimately related to these IR techniques, with a difference – the latter are mostly centered on text indexing and processing, whereas the former are concerned with issues such as query formulation and optimization, indexing and storage strategies, as well as using ontologies to enhance document and query semantics. An example of the latter approach is found in [56], in which ontologies are used to extract information from documents.

Database literature in this area is nevertheless heavily centered on XML documents databases (as opposed to other kinds of documents). As pointed out by [48], there is however a difference between what they call "text-centric XML" – the IR approach, in which information structure is mostly disregarded – and "data-centric XML" (in which the structure is taken advantage of, e.g., in queries or in document correlation). As will be seen, we follow the data-centric approach, since we implement shadows as XML documents stored in an XML database, where queries take advantage of structural relationships among content elements.

2.4 Semantic Web

The Semantic Web is commonly defined as the Web of Data [2]. The main difference between the Web as we know today and the Semantic Web is the focus on the meaning of the data, not only in availability and sharing as before. This information is not related to human consumption, but aims to help machines to understand and consume the information on the World Wide Web [2, 3].

2.4.1 Semantic Annotations

Annotations acquire more semantics when they follow structural schemes and relate concepts and relationships between concepts and/or resources. This strategy allows machine consumption, and therefore the development of new types of applications [28], such as text categorization or multimodal information retrieval.

The concept of Semantic Annotation is derived from the textual annotation concept. Such annotations can have different objectives [40] and be produced and structured in many forms (e.g., links, free remarks, tags, floating layers etc) [14, 31, 6]. Annotations are used, among others, to describe a resource, its relations and what it represents. Informal annotations are usually inserted on documents for human consumption. This hampers computer processing and annotation exchange.

Semantic annotations appeared with the purpose of third-party interpretation, providing explicit and machine interpretable semantics, as supported by Semantic Web standards [28, 14].

As will be seen in Section 5.3, instead of annotating documents, we annotate shadows, thereby concentrating all processing requirements on the shadows, again ensuring independence from specific document formats.

2.4.2 Linked Data and Entity Linking

The notion of Linked Data appeared in the Semantic Web context. The term Linked Data is related to a set of practices for publishing and sharing structured data on the Web. Basically, Linked Data uses the RDF (Resource Descriptor Framework) format to construct typed statements that link things [5, 4]. The 4 “rules” of linked data are: (i) Use URIs as names of things; (ii)use HTTP URIs so that people can look up those names; (iii) when someone looks up a URI, provide useful information; and (iv)include links to other URIs, so they can discover more things [3].

One of the first projects related to Linked Data was DBPedia [2]. The main goal of the DBpedia project is extract structured content from Wikipedia pages, and make them available on the Web. Basically, DBpedia is a RDF dump of the contents produced collaboratively via Wikipedia, allowing the reuse and the exploration of these contents in many other contexts. As will be seen in Chapter 5, we use DBPedia to link Shadow to semantic information.

Linking Open Data[1] (LOD) is a W3C project related to the linked data publishing method. Its main goal is to make several open data sets available and connected on the Web (such as DBPedia, Geonames, WordNet, the GeoSpecies Knowledge Base etc.). To do that, the data sets must publish the data using RDF serialization formats, where URIs link resources on the Web [4]. The LOD project has created a a machine consumable interlinked graph. By September 2010, the project had produced 203 data sets, over 25 billion RDF triples interlinked by around 395 million RDF links. Commonly, DBpedia is described as one of the more famous parts of the Linked Open Data project.

The Linked Data paradigm has made data sharing on the Web easier and enhanced the possibility of aggregating like concepts, creating semantic clusters (e.g., [57]). The same goal is found in the database realm, under the *Entity Linkage* concept, going beyond the interrelationships among Web documents. The idea is to recognize different artifacts that refer to the same real world object, and connect all entities that relate to it, linking them together. This, in turn, allows exploratory queries, and finding out more about an object. Entity linkage is also a strategy proposed for loose integration of heterogeneous data sources.

As pointed out by [26], there are many names under which entity linkage is studied – e.g., deduplication, or entity resolution. Once the linked cluster of entities is constructed, it can be further processed – e.g., eliminating duplication of records, cleaning errors, assigning probabilities to the links (e.g. [26]) or creating graphs that exploitsemantic dependence across linked entities (e.g., [22]). As exemplified in Chapter 5, we process our shadow base under the Linked Data principles, annotating documents via their shadows with DBPedia concepts. Thus, Shadows are used as a means to immerse documents in the Semantic Web.

[1]http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

2.5 Conclusions

The strategy presented in this dissertation is inspired by the concept of *resource descriptors*. *Descriptors* are structures that summarize aspects of some digital object in order to help its indexing, comparison and retrieval [10].

More specifically, our representation strategy borrows from two research fields: image management and metadata standards, described in this chapter. The chapter also points out some basic mechanisms for document management and retrieval, and gives an overview of linked data – all of which will used in the rest of this text.

As discussed in Section 2.2, there are three main strategies related to document processing. Table 2.1 presents a comparison between the three strategies and the SdR approach. For instance, Table 2.1 shows that the SdR approach considers documents' structure and content, preserves original files and offers support to multiple formats, while none of mentioned approaches met these three characteristics.

Strategy	**Characteristics**		
	Consider documents' content and structure	**Preserve original file**	**Support multiple formats**
Documents as general artifacts	no	yes	yes
Conversion to a specific format	yes	no	yes
Interoperable formats only	yes	yes	no
Shadow-driven approach	**yes**	**yes**	**yes**

Table 2.1: Comparison between the SdR approach and the approaches described in Section 2.2

The next chapter presents the SdR – Shadow-driven Representation– proposal itself, discussing and defining the idea behind shadows.

Chapter 3

Shadow-driven Document Representation

This chapter presents the *Shadow-driven Representation* (SdR), a strategy to extract and represent domain relevant elements of documents independently of file formats, and is organized as follows. Section 3.1 gives an overview of the main idea behind shadows. Section 3.2 presents an abstract model and discusses the main concepts related to the SdR approach. Sections 3.3 and 3.4 discuss a strategy to generate shadows and to construct a shadow base, respectively. Finally, Section 3.5 presents conclusions.

3.1 Overview

In this work documents are treated as special cases of complex objects, i.e., they are self-contained units, defining recursive hierarchical containment structures – e.g., a document contains sections, which contain paragraphs, which contain words etc.

The SdR strategy is inspired on the concept of resource descriptors (presented in Section 2.1) and aims to provide transparent[1] support for tasks related to document management, such as indexation, annotation, version control and derivation, and discovery of correlations.

Rather than requiring a specific format or converting the document to a given format to perform some task, the SdR approach proposes an intermediate structure – called *shadow* – which represents key elements of a document. These elements can be defined by users, according to their needs.

[1]Transparency, here, means that the applications that handle the documents do not need to be specialized

An analogy with image descriptors[2] provides an initial context for understanding shadows. First, a shadow isolates a document's format and content from its processing, thus providing a uniform description for a set of documents that can be in many different formats and with a variety of structures. Second, a document can have as many shadows as desired - and thus be (indirectly) processed and queried as wished.

Figure 3.1 presents an abstraction of the main idea behind SdR, where a shadow can represent different document formats, and a document can have different shadows, depending on the domain-user needs. The left side of Figure 3.1 shows a document in some format, while the right side of the figure shows two shadows that represent the same document, but concerning different information.

Still in Figure 3.1, both produced shadows (*Shadow 1* and *Shadow 2* of figure) will be the same, regardless of the original format of the document they represent (e.g., in PDF or ODT[3]). Also, *Shadow 2* is concerned only with features[4] like title, authors and images and their captions, whereas *Shadow 1* describes many other elements.

From a conceptual point of view, shadows can be seen as document descriptors, in the sense that they describe structure and contents of a document according to the specification of a group of users. Figure 3.1 also illustrates the fact that shadows preserve the structure – e.g., Shadow 2 shows that in the document the title precedes the authors, the image precedes the caption and so on.

Shadow Schema and Definition of Elements of Interest

Different domains may have different needs of document handling [24]. To illustrate that, consider a collection of scientific papers in different formats. Users of a domain P_1 may be interested in searching the collection by using parameters like title, authors and keywords. On the other hand, a different domain P_2 may be interested in more specific tasks over the collection, such as processing images inserted in the documents, or even searching for sections labeled as *"Results"* or correlate bibliographic references.

In order to handle these needs, specialized applications are developed and customized. These specialized solutions for P_1 and P_2 will eventually face problems such as "how to perform extraction of elements?", "how to handle format heterogeneity?", "what data structure or database should be adopted?". Again, these questions will require different and specialized solutions.

[2]We could drive the same analogy with sound descriptors, but chose images because of the vast literature in image databases

[3]As presented in Chapter 4, we implemented an API that supports the extraction of elements from textual documents in formats like doc and docx, odt and pdf. That chapter also gives more detailed information about the support for those formats

[4]The term feature here is related to elements of the structure and the content of a document

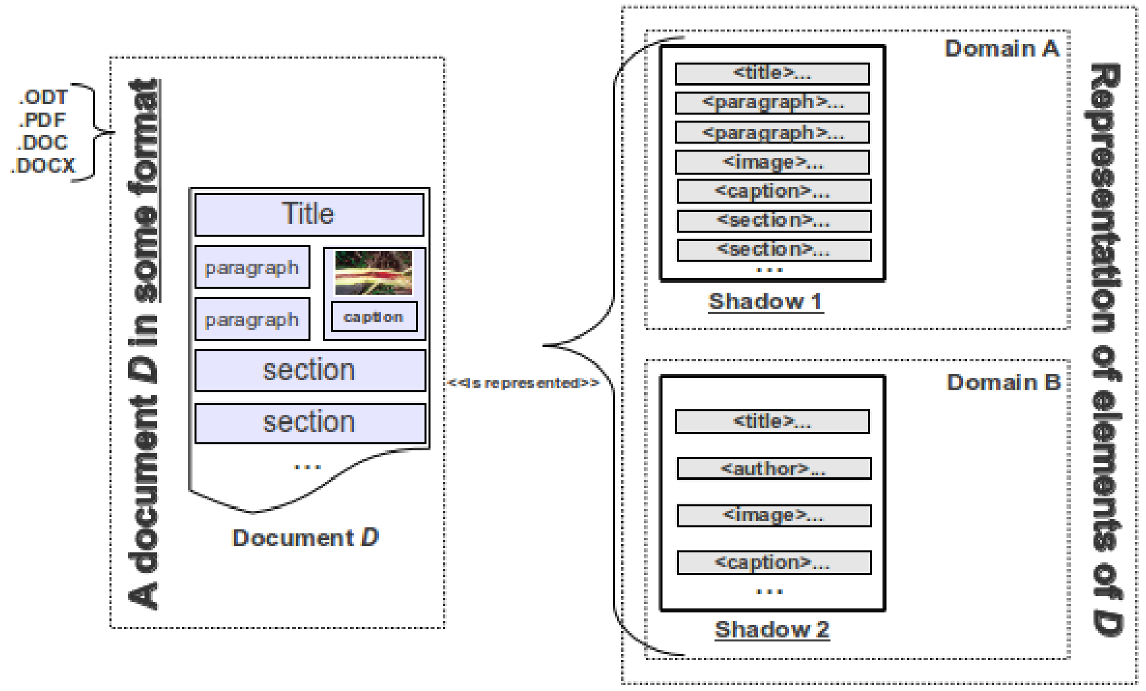


Figure 3.1: Abstraction of the main idea behind the SdR strategy

Continuing the example, in the SdR approach, instead of concerning themselves with specialized solutions, users of both P_1 and P_2 need to define their *elements of interest*. In analogy to databases, we say that these elements define the shadow's *schema*, and that this schema is *instantiated* for the individual documents in the set.

Intuitively, a Shadow Schema specifies which elements should be recognized and represented in the corresponding descriptor (shadow). This definition drives the process of document analysis and shadow instantiation. The possibility of defining different subsets of elements and element levels makes the shadow representation scalable.

In more detail, users define which structural/content elements of a set of documents are relevant to their purposes (e.g., sections, tables, images, captions); and a document collection is processed to construct shadows according to this choice. Thus, a given document can have many shadows (e.g., just as an image can have many descriptors, depending on the features of interest selected).

3.2 Abstract Model

The generation of a shadow is divided in two steps: (a) Definition of elements of interest (the schema); and (b) instantiation of the shadow (for each document in a collection) based on the set of elements of interest defined by users. Stage (b) is organized in two parts: (i) document analysis and recognition of elements of interest; (ii) shadow instantiation.

In the definitions that follow, we adopted database terminology to define shadows, to facilitate understanding the underlying concepts. These are not, however, formal definitions – e.g., our "shadow element types" are not types from a formal point of view. We say that a shadow is a descriptor of a document *under* a schema, since distinct schemas result in different shadows.

Let $D = \{d_1, d_2, d_3, ...d_n\}$ be a set of arbitrary documents (in several formats). For each document $d \in D$, we can define:

Definition 1 *A document d is a tuple $\langle E, H\rangle$ where E is a set of* ***instances*** *of elements within a document, $E = \{e_1, e_2, e_3, \ldots, e_n\}$, and H is a tree structure whose root is the entire document and whose nodes are elements of E.*

Definition 2 *Each* ***recognizable*** *instance $e_i \in d$ is associated with a type $t_\alpha \in T$. $T = \{t_a, t_b, t_c, \ldots, t_\omega\}$ is a set that contains all* ***element types****, such as page, section, chapter, image, table or other element types defined by users*[5].

As presented before, in the SdR approach, users may define elements of interest. This definition is a set of types of elements, giving origin to the **shadow schema**, defined as:

Definition 3 *A* ***shadow schema*** *$\Sigma = (L, J)$ is a tuple that contains a set of labelled element types (L) which users are interested in and the relationship between the types (J). L is a set of pairs associating labels (l) and element types (t) $L = \{l_1 : t_a, l_2 : t_b, \ldots, l_k : t_\omega\}$, where $t_\omega \in T$. J is a tree structure that describes the hierarchical relationship among the elements of L.*

As presented in Figure 3.2, the shadow schema can be seen as a labeled tree of types of elements, corresponding to the elements of interest. These elements are associated with document content (e.g., table, paragraph, list of references) and can be related hierarchically as regards a document's structure (e.g., users may be only interested in captions of images, but not on captions of tables). As highlighted in Figure 3.2, users also can specify that the relation between element types is optional – instances of t_d not necessarily need to contain instances of type t_n.

[5] *Users should be able to specify element types via shadow schema*

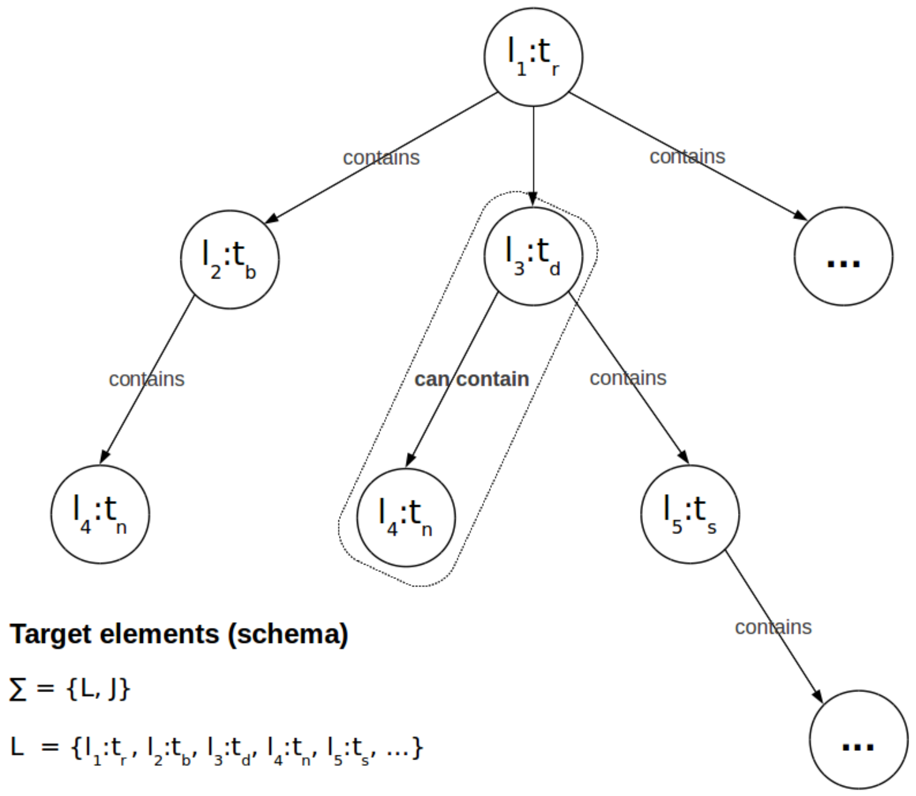


Figure 3.2: Abstraction of a shadow schema and the relationship between the element types. Label l_1 of the schema refers to element of type t_r and so on

As presented in Chapter 4, for implementation purposes, element types are specified via metadata standards and/or namespaces in a pre-defined shadow schema vocabulary, and can preserve an element's hierarchical structure within d. The use of such standards in the schema definition allows processing documents according to distinct domain needs and vocabularies.

Shadow

As mentioned in the introduction to this chapter, this work treats documents as special cases of complex objects [8], i.e., they are self-contained units, defining recursive hierarchical containment structures – e.g., a document contains sections, which contains subsections, which can contain a image or a table etc. Shadows do therefore describe a document's structure and contents.

Since each element instance is associated with a type, we can define:

Definition 4 *given a shadow schema* Σ*, there exists a mapping function* $\delta : d \rightarrow \Sigma$ *capable of relating an* element instance $e \in d$ *with a type* $t \in \Sigma$.

Definition 5 *A* ***shadow*** $S_d(\Sigma)$ *for a document d is a set that contains* ***instances of elements relevant to a domain*** *of a document d, under* Σ.
Given this, a ***shadow*** *can be defined as* $S_d(\Sigma) = \{e_x \in d \mid \delta(e_x) \in \Sigma\}$

It is important to note that different domains may have different needs, implying in different schemas. For instance, in the example presented on the overview, the domain P_1 will define a schema Σ_a and the domain P_2 will define a schema Σ_b. Then, shadows $S_d(\Sigma_a)$ and $S_d(\Sigma_b)$ represent the same document d, but concern different elements of the document's structure and content.

3.3 Shadow Instantiation

Considering the definitions mentioned in previous section, the production of a shadow can be seen as a 4-uple $M = (d, S_d(\Sigma), \Sigma, \delta)$. As presented in Algorithm 1, to produce a shadow $S_d(\Sigma)$ for a document d, basically, an algorithm will need to open the document d and process its content in order to recognize and extract instances of types of elements. Such elements will be represented and persisted into the shadow $S_d(\Sigma)$ using a corresponding label. Shadow construction is performed element-wise, i.e., every instance of an type of element of interest found in d is described in $S_d(\Sigma)$.

Let t_Γ be a type, and $t_\Gamma \in \Sigma$. Let e_i denote an instance of a type t_Γ within document d. Then,

$$S_d(e_i) = e_i \bigvee S_d(e_i) = \mathrm{URI}(e_i)$$

i.e., each instance of an element of interest in a document can be described in the shadow in two ways: the instance can be copied into the shadow using the corresponding label, or be pointed at by the shadow via an URI. As will be seen in the next section, this URI refers to some repository into which the instance e_i is copied for subsequent processing – the choice depends on user requirements.

As we present on Chapter 4, we developed an extractor capable of extracting and recognizing instances of elements of interest and produce shadows according to a given shadow schema.

From an implementation perspective, shadows are specified as a well formed XML document, where element tags reference ontologies or metadata standards – i.e., all tags have semantics.

Since shadows are themselves implemented as XML documents, they can be subsequently processed using, for instance, Information Retrieval (IR) techniques – e.g., to identify string patterns – but they can serve many other purposes.

Algorithm 1 Shadow-Generator(d, Σ, δ)

```
1: read($\Sigma$)
2: open($d$)
3: for all instances $e_i \in d$ do
4:     if $\delta(e_i) \in \Sigma$ then
5:         if the instances should be local then
6:             add $e_i$ to $S_d(\Sigma)$ preserving its hierarchical relationship
7:         else
8:             store $e_i$ in the repository
9:             add URI($e_i$)) to $S_d(\Sigma)$ preserving its hierarchical relationship
10:        end if
11:    end if
12: end for
13: Return $S_d(\Sigma)$
```

3.4 Construction of a Shadow Base

A shadow base is a data repository where shadows are stored. A set of documents D gives origin to one (or multiple) shadow base(s) B, in which shadows can be queried and processed independent of the original document format. In an image base, for instance, images are not ranked or accessed directly – rather, it is the descriptors that are used to rank and access the images.

However, shadows are by construction very different from image descriptors. The fact that they have a schema and contents allows us to construct actual databases of shadows – and thus query shadows using database languages (as presented in Chapter 5). Documents become therefore (indirectly) amenable to database management mechanisms. Second, these database elements are linkable to ontologies – and thus documents become semantically processable. Third, a shadow base can describe local or remote, centralized or distributed sets of documents. Finally, shadows describe arbitrary document elements – not only text, again allowing us to take advantage us database research in document processing. All these issues will be discussed in the following, and exemplified in Section 5.

Finally, as presented in Algorithm 2, a shadow base B for D under Σ is a set $B = \{S_{d_1}(\Sigma), S_{d_2}(\Sigma), \ldots S_{d_n}(\Sigma)\}$.

Abstraction of a shadow base generation

Figure 3.3 illustrates the main steps for constructing a shadow base. First, users define the elements of interest and their hierarchical relationships, thereby specifying the schema. Next, the set of documents is processed by an *Extractor* module that analyzes each

Algorithm 2 Shadow-Base-Constructor(D, Σ, δ, B)

```
1: for all $d_i \in D$ do
2:     $S_{d_i}(\Sigma) \leftarrow$Shadow-Generator($d_i$, $\Sigma$, $\delta$)
3:     add $S_{d_i}(\Sigma)$ to $B$
4: end for
```

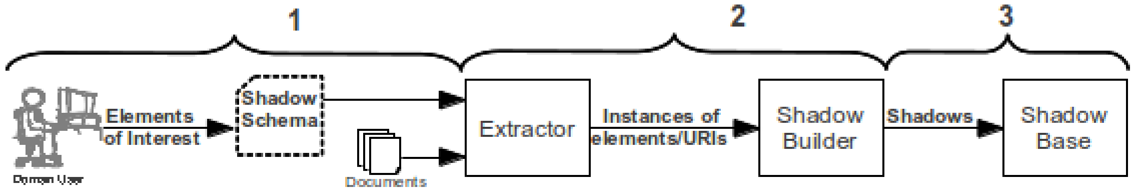


Figure 3.3: Overview of the abstract shadow generation process.

document to recognize the elements of interest, forwarding their instances (or their URI's) to the *ShadowBuilder* module. Finally, the *ShadowBuilder* constructs the shadows and stores them in the shadow base.

The *Extractor* module is a key component in this pipeline. Conceptually, it must be able to process any document format and identify arbitrary elements, themselves defined using arbitrary standards and namespaces. Obviously, it is impossible to construct a universal extractor to satisfy such requirements.

The implementable idea behind the *Extractor* is that it comprises an extensible set of functions that recognizes elements within specific document formats, always obeying the hierarchical definition. In other words, an *Extractor* can "extract" any kind of element within any type of document, as long as code is developed to perform this task. Again using the image descriptor simile, image descriptors are defined mathematically, but the actual implementation varies with image format.

As will be seen next, our extractor implementation already supports[6] documents in three formats – .pdf, .doc and .odt. In other words, our shadows can uniformly summarize documents in these three formats, and recognize a multitude of document element types.

3.5 Conclusions

This chapter presented some details about the SdR proposal, making some definitions and presenting a abstract model to explain the strategy.

[6]It is an initial prototype

The SdR – *Shadow-driven Representation* – strategy is based on building an interoperable document descriptor that summarizes key aspects in a document, allowing its indexing, comparison or annotation.

Shadows can be seen as a generic tree structure that describes a document according to what users identify as elements of interest (e.g., pages, paragraphs, images, sections etc.). As will be seen in Chapter 4, we automatically instantiate shadows using a set of widely adopted metadata standards, after a linear and filetype-sensitive reading – using the DDEx project[7], which was developed by us.

[7]http://code.google.com/p/ddex

Chapter 4

Implementation of a Shadow Generation Process

This chapter discusses implementation issues. Section 4.1 presents an overview of these issues. Section 4.2 presents the technologies adopted. Section 4.3 concerns the architecture of the implemented shadow generation process and presents implementation details. Finally, Section 4.4 presents chapter conclusions.

4.1 Overview

To show the feasibility of the proposal presented on the previous chapter, we implemented the entire shadow generation process.

The implementation of a shadow-generator presents many challenges. The first is the design and development of the extractor, e.g., *"how to perform element identification given the variety of document formats?"*. Another issue concerns shadow instantiation, i.e., *"since shadows should be interoperable, how to instantiate and persist elements and deal with multiple embedded elements inside documents?"*.

Our solution to challenge 1 is based on the separation of document processing (recognition of types of elements of interest) and shadow production, and is better described in Section 4.3.2. We approached the second challenge by constructing shadows as trees that contain the required information about the document and point to external elements extracted from the document (i.e., an image or a table). This tree is serialized as a XML file and stored in a database with native support for XML queries (as presented in Section 4.3.4). With this approach, we were able to use already established technologies and solutions for XML.

This section gives an overview of the big picture of our implementation, presented in Figure 4.1. This solution has some significant differences from the abstract generation

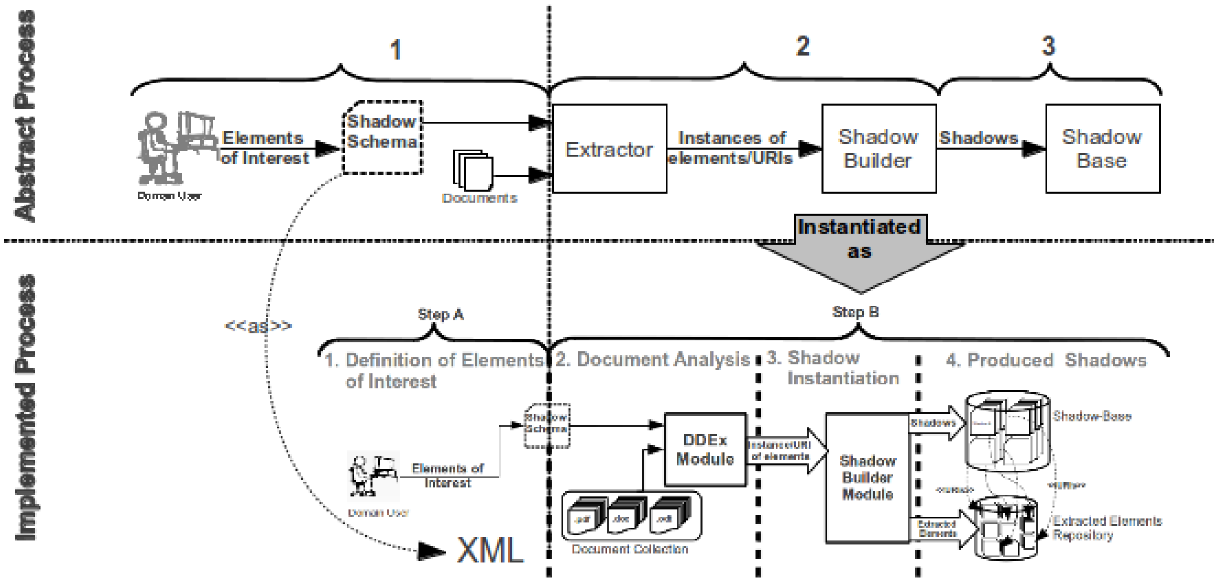


Figure 4.1: Overview of the Implemented shadow generation process and a comparison with the abstract process

process (discussed in the previous chapter with Figure 3.3 repeated for clarity sake).

Our implementation to generate shadows is divided in two main steps: (Step A) definition of elements of interest and schema production; (Step B) shadow generation – based on the elements of interest defined by domain users.

Step B is organized in three parts: The first part (item 2. of Figure 4.1) concerns scanning the document in order to recognize the types of elements of interest; the second part (item 3. of Figure 4.1) concerns the production of shadows, based on the recognized types, according to the schema.

Item 4 in Step B shows that the *Shadow Builder* stores the produced shadows in *Shadow Bases*, but also stores instances of types of elements of interest extracted from documents in a second repository. This repository contains elements from the original document, and are referred to by URIs within a shadow. Such fragments can be for instance a text dump itself or images within a document. These repositories are a means to support the manipulation of parts of documents without interfering with the documents themselves. For instance, we built one repository with images extracted from documents – thereby allowing indexing of these images – and correlating documents according to this kind of content.

4.2 Technologies Adopted

Figure 4.2 reproduces the bottom part of Figure 4.1. The whole process presented in Figure 4.2 was implemented using several technologies, highlighted in red on the figure. For instance, the elements of interest are defined in XML, and documents are processed using pure Java and a set of other APIs specialized in handling documents (listed and described in Section 4.3.2).

As regards *Item 3* of Figure 4.2, we use pure Java and XML parsing and serialization APIs to parse and serialize/persist XML documents (APIs are listed and described in Section 4.3.3).

On the storage side (*Item 4* of Figure 4.2), we store the produced XML files (shadows) in an XML database system, called BaseX[1]. BaseX was chosen since it offers native support for XML query languages, like XQuery [19, 20]. Furthermore, we store pieces of documents (images, tables, references), extracted from the documents, in a (Web)file server, Virtuoso[2]. Virtuoso was chosen because it offers more than a file server, also having a specialized SGBD for RDF triples that provides native support to RDF and the SPARQL query language. As presented in Chapter 5, we connect shadows to some Semantic Web resources via semantic annotations using RDF.

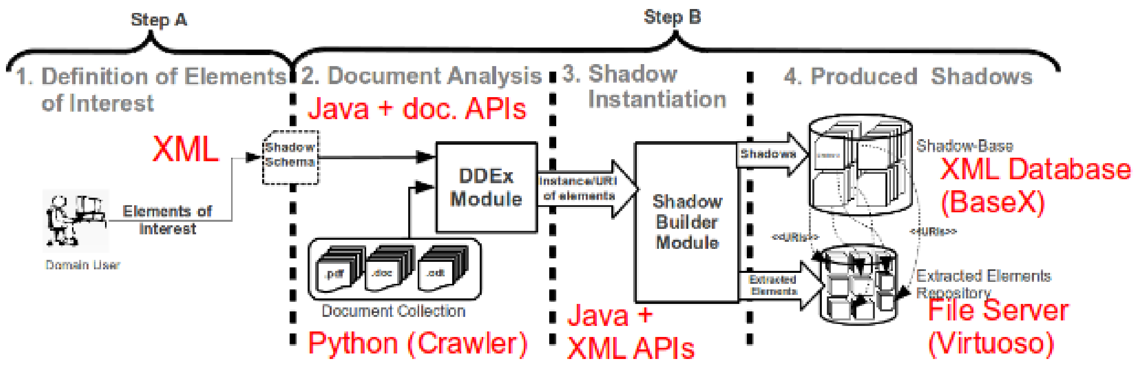


Figure 4.2: Implemented shadow generation process and some of the technologies adopted

[1]http://basex.org
[2]http://virtuoso.openlinksw.com

4.3 System Architecture and Implementation Details

Figure 4.2 gives an overview of the architecture of the implemented solution. In this section, more low-level details about how users can define a shadow schema (*Step A* of Figure 4.2) are presented (Section 4.3.1). Moreover, this section presents further details about each module of the architecture and discuss implementation details about the modules involved in items 2, 3 and 4 of *Step B* of Figure 4.2 (Sections 4.3.2, 4.3.3 and 4.3.4, respectively).

4.3.1 Shadow Schema

As presented before, the Shadow Schema defines which elements are relevant to a specific domain, to be persisted in a shadow. In our implementation, users can specify elements that will compose a shadow by creating a XML file according to a specification.

Basically, this file acts like a template. This template includes all element types, defined via terms of ontologies or metadata standards. In other words, the shadow schema contains namespaces and tags that refer to elements of interest. Furthermore, a shadow schema contains associations, made by users, between types of elements of interest and namespaces and terms. For instance, users interested in the author(s) of a document can define the term *author* from the Dublin Core Standard [11] to represent the author field in a shadow. Alternatively, a user can define his/her own definition of element author. Shadow schemas are defined manually in XML. Future work includes developing tools for shadow schema.

Example of a Shadow Schema in XML

The following XML code presents a piece of the shadow schema used to produce our shadow base in the case study presented in the next chapter. The code can be divided in three parts: (i) Namespaces definition; (ii) types of elements of interest; and (iii) the definition of the relationship between target elements.

Part (i) of the schema is the definition of the valid namespaces in the XML document (according to the W3C namespaces definiton). The root element is <*schematerms:shadowschema/*> (line 4). For instance, line 5 ia property of the root element that defines the Dublin Core Standard as a valid namespace for the shadow.

Part (ii) of the schema concerns the element <*schematerms:targetlist/*> (line 10). It contains the types of elements of interest that should be present in the shadows – e.g., author, number of images, sections, title, captions.

The element <*schematerms:shadowstructure/*> (part (iii) of the code) allows the definition of a hierarchical relationship between the elements previously defined in Part (ii).

For instance, it defines that only sections with images should appear in the shadow. Also, (due to the optional flag) sections with images and tables can be persisted in the correspondig shadow.

```
<?xml version="1.0" encoding="utf-8" ?>

           <!--     Part (i): namespaces     -->
<schematerms:shadowschema
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:stextdoc="http://purl.org/shadow/textdoc"
    xmlns:schematerms="http://purl.org/shadow/schema"
    xmlns:ddex="http://purl.org/ddex"
    xmlns:docbook="http://docbook.org/ns/docbook" >

           <!-- Part (ii): types of elements of interest-->
    <schematerms:targetlist schematerms:instancetype="local">

        <ddex:author/>
        <ddex:title/>
        <ddex:keywords/>
        <ddex:pagecount/>
        <ddex:imagecount/>
        <ddex:wordcount/>
        <ddex:abstract/>
        <ddex:section/>
        <ddex:image/>
        <ddex:caption/>
        <ddex:table/>
        <ddex:references/>

   </schematerms:targetlist>

        <!-- Part (iii): Hierarchical structure-->
   <schematerms:shadowstructure>

        <ddex:section>

            <ddex:url flag="optional" />

            <ddex:image>
                <ddex:caption flag="optional"/>
            </ddex:image>

            <ddex:table flag="optional">
                <ddex:caption flag="optional"/>
            </ddex:table>

        </ddex:section>

   </schematerms:shadowstructure>

            <!-- Continue...->

```

The second part of the schema definition associates an element in the source document to its description in the shadow using association terms. For instance, it shows that a document's title should be described using Dublin Core's (dc) term *title*, and that a section should be described in association with DocBook's term *section*. It is important to note that those associations can be different, depending on the user definition.

```
                <!-- ... Continuing-->

<schematerms:associationlist  >

     <schematerms:elementassociation>

<!--Using term TITLE from Dublin Core for title-->
           <schematerms:elementsource>
                            <ddex:title/>
           </schematerms:elementsource>

            <schematerms:elementmapto>
                              <dc:title/>
           </schematerms:elementmapto>

   </schematerms:elementassociation>

<!--Using term SECTION from DockBook for sections-->
   <schematerms:elementassociation>

            <schematerms:elementsource>
                             <ddex:section/>
            </schematerms:elementsource>

            <schematerms:elementmapto>
                             <docbook:section/>
            </schematerms:elementmapto>

   <schematerms:elementassociation >

<schematerms:shadowschema/>

```

Once a shadow schema is provided, the extractor module (DDEx) will process the document according to this schema. Subsequently, the shadow builder module will map and serialize the extracted elements in a XML document.

4.3.2 Instantiating the Extractor Module – DDEx

One of the main challenges of this work is to deal with the large volume of documents and file formats. Format heterogeneity hampers document processing and consequently the shadow production process. Unlike existing work that converts the document to a specific format or processes only a textual dump of the document, the SdR approach aims to produce a intermediate resource that contains instances of domain relevant elements of a document.

DDEx[3] instantiates the extractor of Figure 4.1. It is a Java framework, implemented by us, that exports instances of elements of documents to applications, using a standard representation, thereby allowing these applications to transparently access the content of documents, regardless of file formats [44].

Figure 4.3 presents an overview on how DDEx is internally organized. Basically, DDEx offers support to handle textual documents (*OpenText* block), spreadsheets(*Open Spreadsheet* block) and presentations (*Open Presentation* block)[4]. Also, DDEx offers a *Util* package that supports a set of common features related to document handling – for instance, a set of features for image handling.

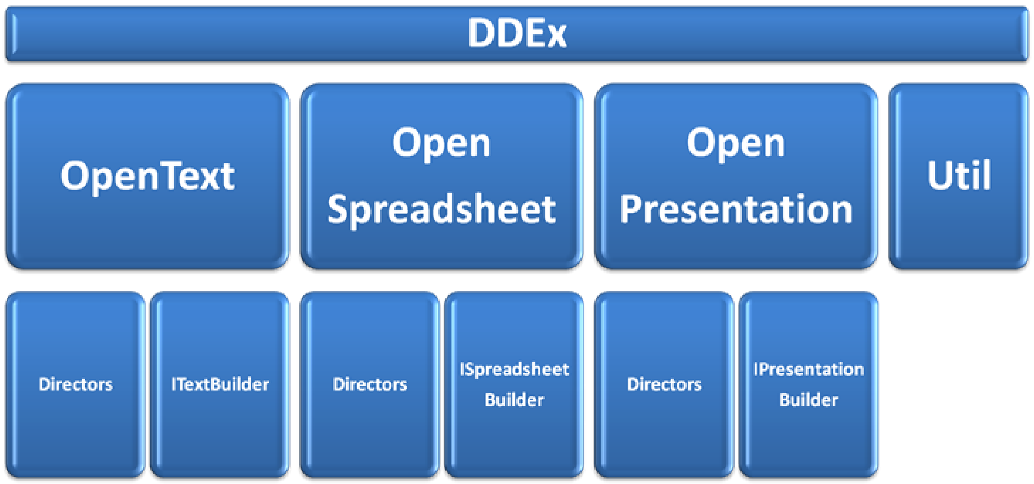


Figure 4.3: Internal organization of the DDEx API

DDEx is implemented according to a specific software design pattern – the *Pattern Builder* [16]. The components of this design pattern can be clearly noted on Figure 4.3, and concern entities:

- **Director**: This is an entity specialized in processing specific files. For instance, the package *Directors* of the *OpenText* block contains a director specialized in processing *.doc* and *.docx* files, another specialized director for *.pdf* files and yet another specialized processor for .odt files.

- **Builder** (*ITextBuilder*, *ISpreadsheetBuilder* and *IPresentationBuilder*): This is a contract (Interface) between an external applications and the directors. This entity is responsible for the separation between external application and the reading of

[3]Open Source Project available at `http://code.google.com/p/ddex`

[4]DDEx was designed to support those types of files, but provides only a poor support for spreadsheet and virtually nothing to presentation files yet

documents by the directors. For instance, a document analyst, implemented by us, implements this interface in order to receive the elements extracted from the specialized directors and process it according to a given schema.

DDEx adopts several APIs for document handling, such as *iText*[5], *PDFBox*[6], *PDF-Clown*[7] and *PDF Renderer*[8] for PDF documents. In case of documents produced in *Microsoft Word*, DDEx adopts the *Apache POI*[9] framework. Furthermore, DDEx adopts the *ODF Tool Kit*[10] and JOpendocument[11] for files following the *Open Document Format.*

Each specialized document processor (director) module within DDEx works as a back-end module, which recognizes elements from the document's content and implements a standard output API able to produce a sequential stream of descriptive calls, reflecting the document internal structure and content.

The Document Analysis and Content Extraction Process

Figure 4.4 gives a high level overview of the Content Extraction process. A set of format-specialized processors (extreme left of the figure) identifies and extracts instances of elements from the documents (using specialized processors from DDEx), forwarding them to a *Document Analyst.*

The analyst is format independent, and processes elements according to a shadow schema. It forwards instances of elements as descriptive[12] calls to the *Shadow Builder* module (more details about the *Shadow Builder* module are presented in next section). Those calls are used to build a shadow, independent of document format. Examples of calls include *call(foundSection)* or *call(foundMultimediaObject)* – where object information and a byte stream of the object itself is transferred to the *Shadow Builder.*

Figure 4.5 abstracts an important concept in this work: composition. The *Document Analyst* is responsible for the task of analysing the composition of elements according to the schema.

The left part of Figure 4.5 shows an abstraction of a specification of an element. This element, whose composite type is *Image*, is defined as follows. An image contains a picture and a caption. A Picture contains a byte stream[13] of the image file. A Caption contains a

[5]http://itextpdf.com

[6]http://pdfbox.apache.org

[7]http://www.stefanochizzolini.it/en/projects/clown/index.html or http://sourceforge.net/projects/clown/

[8]http://java.net/projects/pdf-renderer/

[9]http://poi.apache.org

[10]http://odftoolkit.org/projects/odfdom

[11]http://www.jopendocument.org/

[12]The call is always generic, but contains an object (which contains properties) capable of describe the instance

[13]The raw set of bytes that correspond to the image

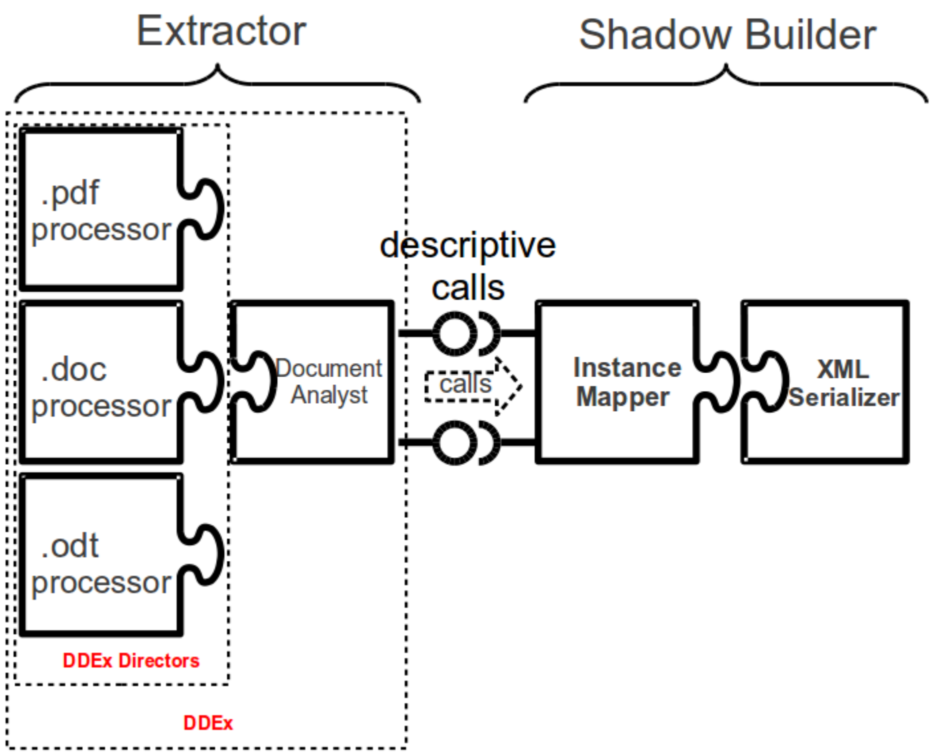
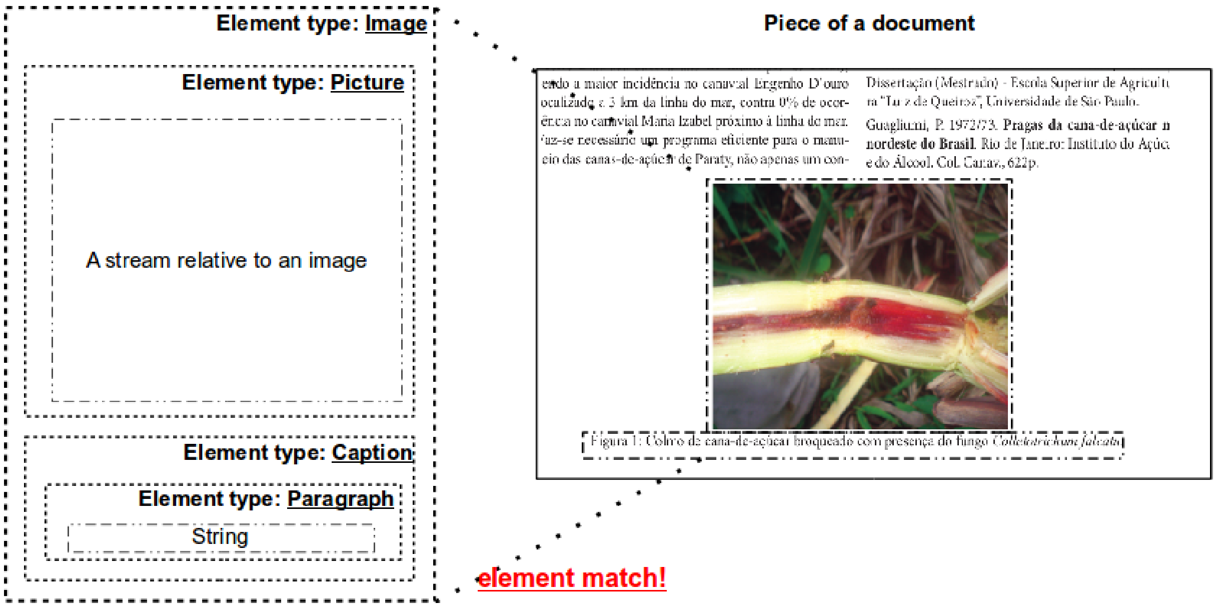


Figure 4.4: Overview of modules involved in the analysis of document content analysis and shadow production

Paragraph. A Paragraph contains a string and a newline command, and a Caption should appear below a Picture.

The right side of Figure 4.5 shows a specific part of a document that fits this specification of many levels of composition. Since this part of the document "matches" the specification (defined in the shadow schema), it should be forwarded to the shadow builder as a descriptive call.

Our decision to build DDEx based on the *Pattern Builder*, and build the *Document Analyst* using DDEx, ensure extensibility of the supported document formats. Since the Pattern Builder allows the separation between the extraction and the production process, we were able to take advantage of the three specialized document processors – for *.doc*, *.pdf* and *.odt* files. Hence, the Shadow Builder does not need to change even if DDEx provides support to new formats of documents.



endo a maior incidência no canavial Engenho D'ouro ocalizado a 3 km da linha do mar, contra 0% de ocorência no canavial Maria Izabel próximo à linha do mar. 'az-se necessário um programa eficiente para o manuseio das canas-de-açúcar de Paraty, não apenas um con-

Dissertação (Mestrado) - Escola Superior de Agricultura "Luiz de Queiroz", Universidade de São Paulo.

Guagliumi, P. 1972/73. **Pragas da cana-de-açúcar n nordeste do Brasil.** Rio de Janeiro: Instituto do Açúc e do Álcool. Col. Canav., 622p.

Figura 1: Colmo de cana-de-açúcar broqueado com presença do fungo *Colletotrichum falcatu*

Figure 4.5: Example of how the document analyst recognizes an image with a caption via type matching of composite elements

Document Pre-processing and Noise Removal

Before proceeding with the analysis and extraction, we pre-process documents in order to identify non-supported formats and malformed documents. Furthermore, pre-processing is needed to recognize the minimum and maximum font size, metadata, number of pages, number of images and other metadata.

The pre-processing phase aims to avoid the following problems:

- **Incomprehensible character encoding and malformed documents :** Some documents can be corrupted or may contain unknown charsets. In addition, some documents can be corrupted.

- **Document as images:** Some formats (mostly PDF) are used to store images. Since we do not support Optical Character Recognition (OCR), our implementation avoids this category of documents because is not possible to recognize elements of interest.

- **Documents with content protection:** Some formats allow the user to protect document contents. We discard protected documents since it is not possible to have full access to the document content.

4.3.3 Shadow Builder Module

Shadows are created as well formed XML files (Item 3 of Step B of Figure). In order to do this, we adopted the JDOM[14], DOM4J[15] and Xerces[16]Java APIs to parse and manipulate and serialize XML files.

Figure 4.6 gives an overview of the *Shadow Builder Module* and how it works. Basically, the Shadow Builder receives a call that contains a piece of the given schema and the corresponding instance of content/structure previously extracted from the documents by the extractor module (DDEx).

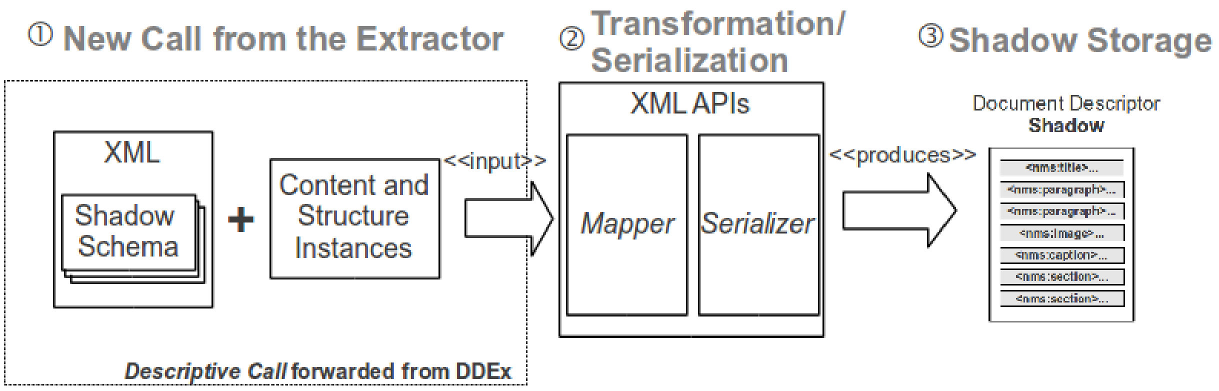


Figure 4.6: Overview of the shadow builder module

Inside the Shadow Builder Module itself (Item 2 of Figure 4.6), there are a *mapper* and a *serializer* unit. The mapper receives instances from the *Document Analyst*, implemented inside DDEx, as a descriptive call that contains: (i) a tree that preserves the relationship between the elements; and (ii) information about and the instance itself.

After receiving a new call corresponding to a new instance found, the mapper instantiates a corresponding element (and its properties) in a tree allocated in memory, preserving its hierarchical relationships. This new element (and its related elements) follows the user definitions on the shadow schema. For instance, if users define that the Dublin Core Term *author* should be used to refer to the author of a document, the mapper defines this on adding this element to the tree. Alternatively, as discussed before, elements also can be stored apart from the shadow itself. After the end of the process, the serializer will materialize the XML document and store it.

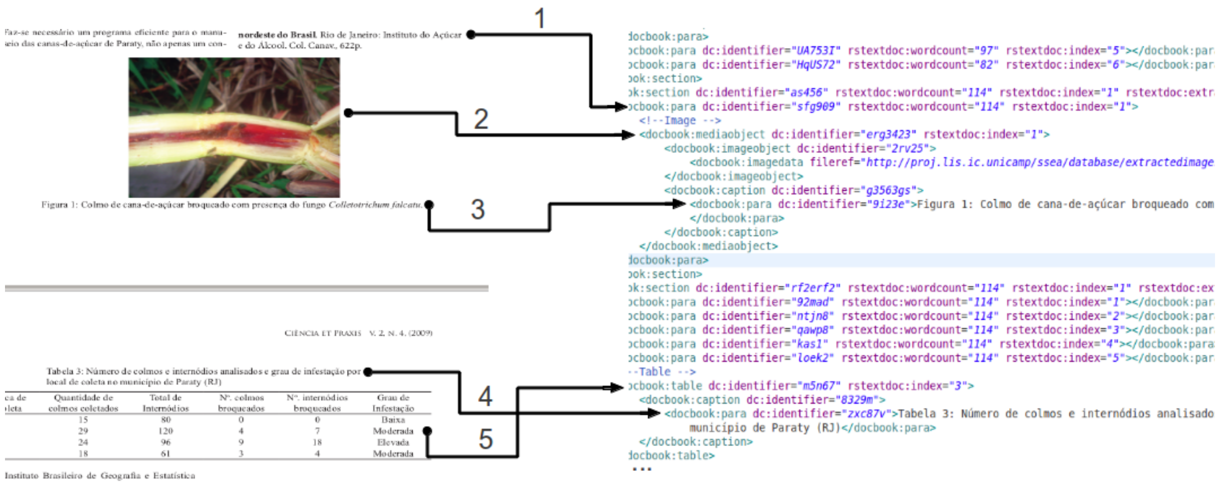
[14]http://www.jdom.org
[15]http://dom4j.sourceforge.net/
[16]http://xerces.apache.org/xerces-j/

4.3.4 Shadow Production and Storage

The Shadow Builder constructs the shadow according to a schema. For each element instance/URI forwarded by the extractor (see Figure 4.1), it builds an XML expression that represents an *instance* or URI of the instance.

For instance, if the shadow schema specifies that sections in a document should be mapped as the term *section* from the DocBook [52] standard, when a call *foundSection* is invoked, it is instantiated in the shadow as the DocBook element *section*. This is not a purely sequential process, because of the hierarchical structure of a schema (e.g., a section can contain a table that can contain a figure).

Table 4.1 shows the set of metadata standards and ontologies already supported by our implementation. Its first line, for instance, indicates that the *Docbook* standard is supported – with the prefix docbook – in our implementation.



Faz-se necessário um programa eficiente para o manu-
seio das canas-de-açúcar de Paraty, não apenas um con-

nordeste do Brasil. Rio de Janeiro: Instituto do Açúcar
e do Álcool. Col. Canav., 622p.

Figura 1: Colmo de cana-de-açúcar broquedo com presença do fungo *Colletotrichum falcatu.*

Ciência et Praxis V. 2, N. 4, (2009)

Tabela 3: Número de colmos e internódios analisados e grau de infestação por local de coleta no município de Paraty (RJ)

| ca de
leta	Quantidade de colmos coletados	Total de Internódios	Nº. colmos broqueados	Nº. internódios broqueados	Grau de Infestação
	15	80	0	0	Baixa
	29	120	4	7	Moderada
	24	96	9	18	Elevada
	18	61	3	4	Moderada

Instituto Brasileiro de Geografia e Estatística

```
docbook:para>
ocbook:para dc:identifier="UA753I" rstextdoc:wordcount="97" rstextdoc:index="5"></docbook:par
ocbook:para dc:identifier="HqUS72" rstextdoc:wordcount="82" rstextdoc:index="6"></docbook:par
ook:section>
ok:section dc:identifier="as456" rstextdoc:wordcount="114" rstextdoc:index="1" rstextdoc:extr
ocbook:para dc:identifier="sfg909" rstextdoc:wordcount="114" rstextdoc:index="1">
  <!--Image -->
  <docbook:mediaobject dc:identifier="erg3423" rstextdoc:index="1">
      <docbook:imageobject dc:identifier="2rv25">
          <docbook:imagedata fileref="http://proj.lis.ic.unicamp/ssea/database/extractedimage
      </docbook:imageobject>
      <docbook:caption dc:identifier="g3563gs">
          <docbook:para dc:identifier="9i23e">Figura 1: Colmo de cana-de-açúcar broqueado com
          </docbook:para>
      </docbook:caption>
  </docbook:mediaobject>
docbook:para>
ook:section>
ok:section dc:identifier="rf2erf2" rstextdoc:wordcount="114" rstextdoc:index="1" rstextdoc:ex
ocbook:para dc:identifier="92mad" rstextdoc:wordcount="114" rstextdoc:index="1"></docbook:par
ocbook:para dc:identifier="ntjn8" rstextdoc:wordcount="114" rstextdoc:index="2"></docbook:par
ocbook:para dc:identifier="qawp8" rstextdoc:wordcount="114" rstextdoc:index="3"></docbook:par
ocbook:para dc:identifier="kas1" rstextdoc:wordcount="114" rstextdoc:index="4"></docbook:para
ocbook:para dc:identifier="loek2" rstextdoc:wordcount="114" rstextdoc:index="5"></docbook:par
--Table -->
ocbook:table dc:identifier="m5n67" rstextdoc:index="3">
  <docbook:caption dc:identifier="8329m">
      <docbook:para dc:identifier="zxc87v">Tabela 3: Número de colmos e internódios analisado
          município de Paraty (RJ)</docbook:para>
  </docbook:caption>
docbook:table>
...
```

Figure 4.7: Piece of a document and the corresponding shadow

Figure 4.7 presents a piece of a document and the corresponding shadow. This example shows that the schema supports types from distinct standards. The code shows that the shadow adopts the *docbook* standard to describe paragraphs within a section (first arrow), caption within the description of a figure (third arrow), table caption and table (fourth and fifth arrows) and a media element (second arrow). It also adopts others standards, such as DCMI Type Vocabulary[17] standard from Dublin Core and the ORE standard. As part of the image descriptor, there is a link to a local repository in to which the image was imported, so that can afterwards be processed apart from the document. Each element

[17]http://dublincore.org/documents/dcmi-type-vocabulary/

Prefix	**Namespace URI**	**Description**	**Role in the Shadow**
docbook	http://docbook.org/ns/docbook	Semantic markup language for representing documents	Presentation-neutral solution that captures and represents the logical structure and content of a document
dc	http://purl.org/dc/elements/1.1/	Set of general metadata terms, used to describe resources	Subset of the adopted terms used to represent documents' metadata
dcterms	http://purl.org/dc/terms/	Additional and refined Dublin Core terms	Another subset of adopted terms used to represent documents' metadata
ore	http://www.o ... s.org/ore/terms/	A standard for description and exchange of aggregations and relations between resources	Used to represent relations between elements of a document
rdfs	http://www.w3.o ... rdf-schema#	RDF Schema vocabulary	Used for typing and general element labelling
adobe-xmp	http://ns.adobe.com/xap/1.0/	Set of metadata terms proposed by Adobe	Another set of adopted metadata terms

Table 4.1: Comparison between the SdR approach and the approaches described in Section 2.2

in the shadow receives an id assigned by DDEx, which will uniquely identify it within the Shadow Base – e.g. arrow 5 points to a table uniquely identified as "*m5n67*" via the *identifier* Dublin Core term. Furthermore, while some elements are referred to by links within the shadow (e.g., the table), others are copied into the shadow (e.g, image caption).

As pointed out by [54], finding good structures to represent documents is complicated because of the high dimensionality of documents. Here, since shadows are themselves documents, we did not solve the dimensionality problem. However, since they describe a document according to a given set of elements of interest defined by users, the dimensions are more controlled, and thus more manageable.

4.4 Conclusions

This chapter presented the a software architecture to produce shadows and store them. This is an instantiation of the process presented in Chapter 3.

This chapter also presented more low-level details on each step of the proposed architecture, discussing adopted technologies and implementation details.

Chapter 5

Case Study

This chapter presents a case study where shadows are used to pose queries over the content and structure of documents, and to indirectly annotate documents. This case study is divided in three parts, and goes from the construction of a shadow base, concerning documents related to the biodiversity domain, to the use of shadows to produce semantic annotations for documents.

Section 5.1 corresponds to the first part of the case study, and details our effort to create a shadow base (using the implementation presented in Chapter 4). Section 5.2 corresponds to the second part of the case study, and discusses how we used shadows to pose unstructured and structured database queries against a document's content and structure. The third part of the case study is presented in Section 5.3, discussing details about our strategy to link (via semantic annotations) shadows to linked data resources. Finally, Section 5.4 presents conclusions.

5.1 Part I: Construction of a Shadow Base

Constructing a Shadow Base of Documents Concerning Biodiversity Studies

Our case study concerns papers in Portuguese and English for biodiversity studies. To build a document collection, we implemented a Python crawler to automatically get documents from Google Scholar[1][2], constructing a collection of approximately 3200 documents. In addition, we also included master dissertations, PhD theses and several papers published by members of our lab, totaling 3300 documents. The documents are uniquely identified and stored in a repository, occupying 1.9 GBytes.

[1]`http://scholar.google.com`

[2]Keywords used on the search where: Biodiversity, Biology, Biodiversidade, Biologia

From this document collection (textual documents in *.doc* and *.docx*, *.pdf*, *.odt* formats), our implementation generated 3104 shadows. 196 documents could not be processed due to corrupted files, document as images and unknown file formats.

Documents of this collection had 2053 images extracted, which were stored in a repository reachable in the web and uniquely identified (URL). Shadows were identified with the same name of the corresponding document, and also became available on the Internet, thus having an URI – for instance, the shadow in *http://proj.lis.ic.unicamp.br/ssea/database/shadows/64.pdf.xml* is a description of the document in *http://proj.lis.ic.unicamp.br/ssea/database/documents/64.pdf* according to the schema adopted[3].

Part 1

```
<?xml version="1.0" encoding="utf-8" ?>
<schematerms:shadowschema xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:stextdoc="http://purl.org/shadow/textdoc"
  xmlns:schematerms="http://purl.org/shadow/schema"
  xmlns:ddex="http://purl.org/ddex"
  xmlns:docbook="http://docbook.org/ns/docbook">

        <schematerms:targetlist schematerms:instancetype="local">
                <ddex:author />     <ddex:title />    <ddex:keywords />
                <ddex:imagecount /> <ddex:abstract /> <ddex:section />
                <ddex:image />      <ddex:caption />  <ddex:table />
                <ddex:references />
        </schematerms:targetlist>
```

Part 2

```
        <schematerms:shadowstructure>

                <ddex:metadata>
                        <ddex:author /> <ddex:title /> <ddex:keywords /> <ddex:imagecount /> <ddex:wordcount />
                </ddex:metadata>
                <ddex:abstract />
                <ddex:chapter flag="optional">
                        <ddex:section>
                                <ddex:paragraph />
                                <ddex:table flag="optional">  <ddex:caption />  </ddex:table>
                                <ddex:image flag="optional">  <ddex:caption />  </ddex:image>
                        </ddex:section>

                </ddex:chapter>

        </schematerms:shadowstructure>
```

Part 3

```
        <schematerms:associationlist>

                <!--Using term TITLE from Dublin Core for title -->
                <schematerms:elementassociation>
                        <schematerms:elementsource>  <ddex:title />  </schematerms:elementsource>
                        <schematerms:elementmapto>   <dc:title />    </schematerms:elementmapto>
                </schematerms:elementassociation>

                <!--Using term SECTION from DockBook for sections -->
                <schematerms:elementassociation>
                        <schematerms:elementsource>   <ddex:section />    </schematerms:elementsource>
                        <schematerms:elementmapto>    <docbook:section /> </schematerms:elementmapto>
                </schematerms:elementassociation>
        </schematerms:associationlist>
     ...
```

Figure 5.1: Shadow schema (XML) adopted in the case study

Figure 5.1 presents an excerpt (written in XML) of the shadow schema that guided the

[3]The decision to store both shadows and documents in Web repositories, thus giving them a valid URI, was based on the the rules of linked data, as presented in Section 2.4.2

construction of the shadow base. Highlighted Part 1 of the figure defines the adopted namespaces and elements of interest. Part 2 of the figure defines the hierarchical relationship between the elements of the interest – e.g., a section can contain a table or an image, and both tables and images must contain a caption – those relationship can be better visualized via Figure 5.2. Part 3 of the figure defines how elements should appear in the generated shadow – e.g., it defines that the title of the document should be persisted usind the term *title* from the dublin core, and sections should appear in the shadow as the term *section* from DocBook.

On top of Figure 5.2 a shadow schema is abstracted, listing the element types that we are interested in and their hierarchical relationships – for instance, we define that tables should have a caption. The bottom part of Figure 5.2 shows the terminology adopted – as defined in the XML of part 3 of Figure 5.1, the title of the document should be instantiated as an element *title* from the Dublin Core metadata initiative [11], and a section should be persisted with the term *section* from DocBook [51].

The shadow base generated according to this schema was stored and loaded in the BaseX XML DBMS. It occupies 502 MB. The shadows were created in 26 minutes in a computer with 16 processors and 32 GB of RAM, using a 64bit linux distribution.

5.2 Part II: Querying the Shadow Base

We formulated queries in Xquery [12] against the BaseX shadow base (approximately 3000 shadows stored), following end-user (biologist) requirements. We also executed queries that were not relevant to biologists, but which show the potential of our proposal (as compared to IR approaches, or document management proposals).

Figure 5.3 is an example of a query result that can be used to analyze the document collection via the shadow base, though it is not a query requested by our end-users. It shows the shadow base (as displayed by BaseX) in terms of documents that contain a number X of images.

The following code is the XQuery that produced the result presented in Figure 5.3. Both lines 15 and 16 concern the definition of the namespaces that will be used on the query. Line 18 is the query itself, a XPath expression that points to an element of the shadow that contains the number of images within a document.

```
16 declare namespace docshadow  = "http://purl.org/documentshadow";
17 declare namespace textshadow = "http://purl.org/textshadow";
18
19 /docshadow:shadow/docshadow:metadata[textshadow:imagecount>X]
```

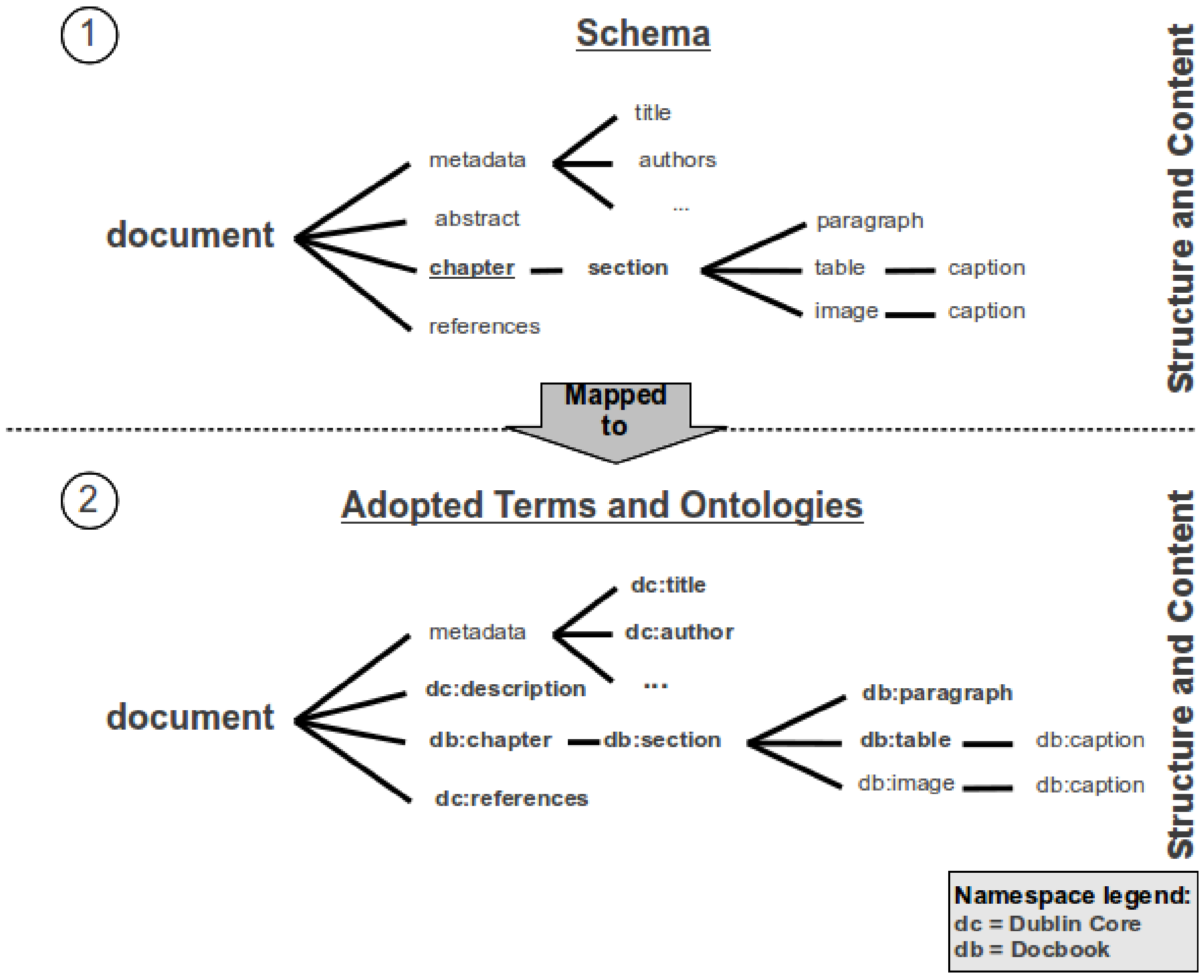


Figure 5.2: Abstraction of the shadow schema used in the case study

Figure 5.3a shows that a large number of shadows (and therefore documents) contain images, whereas 5.3d shows that very few documents contain more than 100 images.

Examples of the latter include a technical report on analysis of vegetation cover in agricultural regions in Brazil, using satellite images, a book chapter on Computational Biology and a paper discussing parasites that compete for host species. This is an example that shows that the shadow base can be queried to partition the document collection according to several criteria - and this criterion (number of images), in particular, is not viable in other approaches.

Examples of queries that can be processed by IR techniques as well include:

Q1 Documents whose authors include researcher called ”A” and whose title contains the word ”biodiversity”.

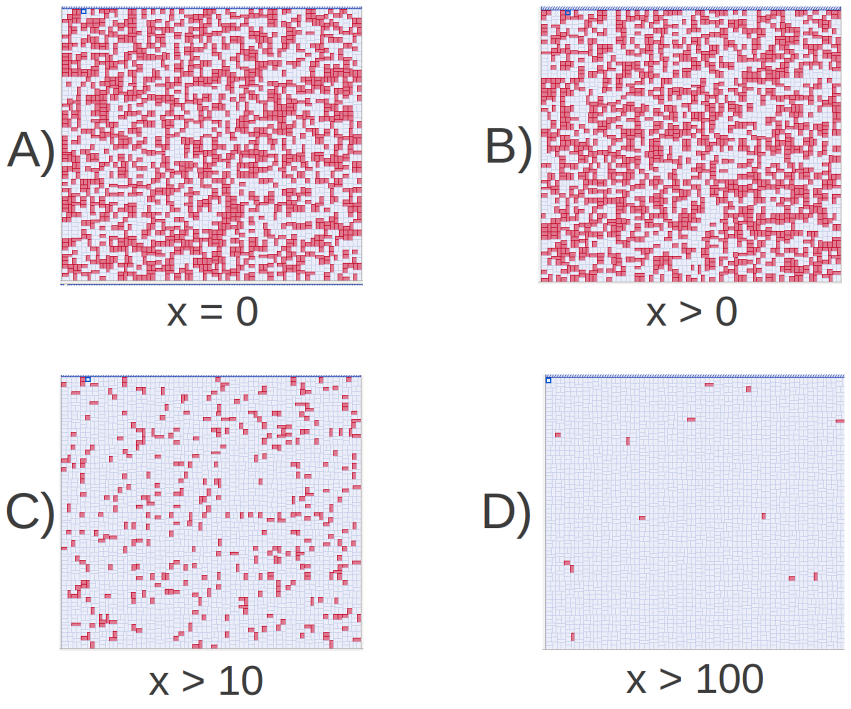


Figure 5.3: Fetched shadows from a query “documents with x images”

Q2 Documents containing keywords ”C” and ”D”.

For instance, query Q1 returned 17 shadows (corresponding to *.pdf* and *.doc* files) for “A”= “Marcelo” and 8 shadows for “A” = “Vera”. Query Q2 returned 33 shadows for “C” = “monitoring” and “D” = “report”.

Examples of queries that cannot be processed by other approaches include:

Q3 Documents with more than 10 pages, at least 5 images, and one section called ”Case Study”.

Q4 Documents without images or tables.

Queries Q3 and Q4 above combine structure and content. They are not useful in terms of biodiversity research, but are included to show the versatility of shadow management.

Q5 Documents that contain one or more images of species ”Glomerella tucumanensis”.

This query is posed against image captions, which are part of the schema and associated with images under the shadow *image* element.

Q6 Documents that contain an image of "Glomerella tucumanensis" followed by a table that concerns the same species.

This query is posed against shadow image and table captions.

5.3 Part III: Annotating Documents via Shadows

As presented before, using a database language for XML queries, like XQuery, we were capable of (indirectly) posing queries about structure and content of a collection of documents. Nevertheless, queries on the shadow base are limited to the data extracted from the documents.

Since a shadow is an interoperable document descriptor, it can be easily used with different purposes. In our case study, we decided to use shadows to indirectly annotate documents, creating links between elements of a shadow and ontologies and other semantic Web resources – producing semantic annotations.

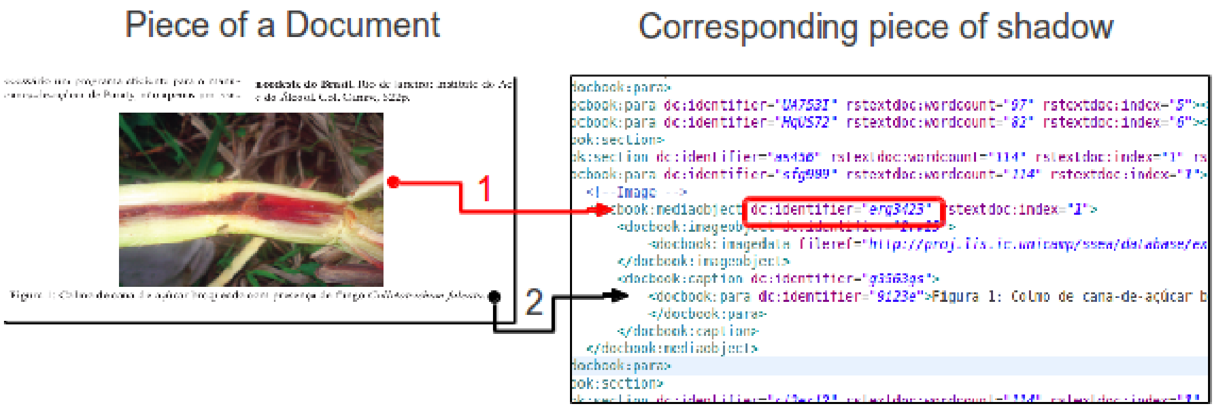
To annotate shadows, we adopted an RDF-based schema for describing annotations and an established XML reference standard to address shadow elements. There are several standards/languages to refer and link XML documents, such as shadows. Those W3C standards – such as XPath, XLink and XPointer [55] – provide a set of tools that allows the addressing of XML documents and their fragments. XPath models an XML document as a tree of nodes and provides a URL path notation for element addressing, while XLink allows elements to be inserted into the XML documents to create and describe links between resources. Finally, XPointer defines a language to be used to locate a fragment via a URI, allowing a URI reference to locate some resource.

5.3.1 Producing Semantic Annotations

Our annotation strategy is based on the Linked Data paradigm. It follows the simple strategy of considering that two entities are the same if they refer to the same ontology term – i.e., we do not look for more sophisticated IR techniques. The goal is to establish a basis for fact finding and linking documents and concepts via shadows, assuming that the two entities – the element instance in a document, and the concept defined in the ontology – are semantically related. This is a strong assumption (e.g., see [22]), but it is the first step towards semantic entity linkage in a context of otherwise heterogeneous unrelated data sources.

For the purpose of this experiment, we annotated shadows manually. Future developments will incorporate our work on semi-automatic annotation processes, guided by workflows [33]. Annotations were inserted in a Virtuoso[4] database, that supports RDF triples and SPARQL queries[5]. In more detail, our annotations are RDF triples that link a shadow element to concepts in the LOD.

To illustrate how elements of a document can be indirectly linked to concepts in the LOD (presented in Section 2.4.2), consider Figure 5.4. Left side of the figure shows a specific part of a document that contains an image of a fungus (*Colletotrichum falcatum*, also named as *Glomerella tucumanensis*). Arrow 1 points to a specific element of the XML that represents the figure, uniquely identified as "erg3423". Arrow 2 of the figure is pointing to another element (within the element pointed by arrow 1) that corresponds to the caption of the figure within the document (left side).



```
docbook:para>
ocbook:para dc:identifier="UA753I" rstextdoc:wordcount="97" rstextdoc:index="5"><
ocbook:para dc:identifier="HqU572" rstextdoc:wordcount="82" rstextdoc:index="6"><
ook:section>
k:section dc:identifier="as456" rstextdoc:wordcount="114" rstextdoc:index="1" rs
ocbook:para dc:identifier="sfg999" rstextdoc:wordcount="224" rstextdoc:index="7">
  <!--Image -->
  <docbook:mediaobject dc:identifier="erg3423" rstextdoc:index="1">
    <docbook:imageobject dc:identifier="[illegible]">
      <docbook:imagedata fileref="http://proj.lis.ic.unicamp/ssen/database/ex
    </docbook:imageobject>
    <docbook:caption dc:identifier="g3563gs">
      <docbook:para dc:identifier="9123e">Figura 1: Colmo de cana-de-açúcar b
      </docbook:para>
    </docbook:caption>
  </docbook:mediaobject>
docbook:para>
ook:section>
```

Figure 5.4: A captioned image within a document (left part) and a corresponding shadow XML code (right part)

The piece of SPARQL code in Figure 5.5 shows how the specific part of the document (shown in the left side of Figure 5.4) is (indirectly) linked with the concept of the fungus *Glomerella tucumanensis* in the LOD. The idea is link the specific element of the shadow (identified as *dc:identifier="erg3423"* in the right side of Figure 5.4) to concepts in the LOD (specifically, with the concept of the *Glomerella tucumanensis* in the DBPedia).

The link between the shadow and the concept is performed by an insertion of an RDF triple in the Virtuoso database, and is structured as

[4]http://virtuoso.openlinksw.com
[5]Our database is open, and a SPARQL service can be found at `http://proj.lis.ic.unicamp.br/sparql`

```
INSERT into graph (of the annotation database)

 resource --- URI

 property --- depiction

 value --- the element (i.e., image in shadow 64.pdf.xml)
```

and links (via *foaf:depiction* of Figure 5.5) the DBPedia URI for "Glomerella tucumanensis" to the specific shadow in the Shadow Base. More specifically, this link is associated with an element identified within this shadow by "erg3423" – see last line of the code in Figure 5.5. We recall that this identifier is artificially generated by DDEx to support the management of shadow elements within the Shadow Base.

```
PREFIX foaf: <foaf:>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

INSERT IN GRAPH

 <http://proj.lis.ic.unicamp.br/annotations/>

{
<http://dbpedia.org/resource/Glomerella_tucumanensis>

<foaf:depiction>

<http://proj.lis.ic.unicamp.br/ssea/database/shadows/64.pdf.xml#
xpointer(dc:identifier('erg3423'))>
}
```

Figure 5.5: SPARQL code to create a nannotation

5.3.2 Querying Annotations

The next piece of code in Figure 5.6 shows a query for URIs and values associated with the DBPedia concept "Glomerella tucumanensis". This query is posed against the local Virtuoso instance, which contains the annotations and data imported from DBPedia, Geonames and Geospecies. Here,the result is very different from just using XQuery to query shadows. The first difference, of course, is that the query in XQuery retrieved URIs of shadows, and this query retrieves URIs of shadows and of other concepts.

```
SELECT ?property, ?value


WHERE {

<http://dbpedia.org/resource/Glomerella_tucumanensis>

   ?property

   ?value

}
```

Figure 5.6: SPARQL code for querying the Virtuoso database

The second difference is semantically more interesting. While the (Xquery) query Q6 on the Shadow Base in Section 5.2 only returned shadows of documents that had images of this species, resulting in 3 shadows, the (SPARQL) query of Figure 5.6 on annotations plus LOD data returned 46 URIs, of which 3 point to the same shadows, and one to *another* shadow, not identified by the Shadow Base query. The reason for this discrepancy is the following. "Glomerella tucumanensis" has several scientific synonyms - one of them being "Colletotrichum falcatum". Thus, the SPARQL query on annotations not only returned the shadows retrieved using XQuery; it also returned references to a shadow (and thus a document) that had no mention of "Glomerella", but described it under another name.

Table 5.1 shows an excerpt of the 46 answers to the SPARQL query shown in Figure 5.6 – the URIs of all resources related to "Glomerella tucumanensis". Not only does it return the document that have been annotated by the code in Figure 5.5 in its 6th row, but also returns links to several other DBPedia information and resources.

In particular, in biodiversity studies, experts need to correlate papers in several scientific domains (e.g., climatology, phenology, pedology), For instance, continuing the example related to the Figure 5.4, "Colletotrichum falcatum" is a fungus that attacks sugar cane and is widespread in subtropical regions, being also called "red rot of sugar cane". Via annotations, experts are able to pose queries such as "*documents that refer to plant diseases in subtropical regions*" or "*documents that contain images of red rot of sugar cane*". This is an example of how annotations using LOD can significantly enhance

Property	**Value**
rdf:type	dbpedia:Species
rdf:type	dbpedia:Fungus
rdf:type	dbpedia:Eukaryote
rdfs:label	”Glomerella tucumanensis”@en
rdfs:comment	”Glomerella tucumanensis is a plant pathogen.”@en
foaf:name	”Glomerella tucumanensis”@en
dc:subject	http://dbpedia.org/resource/Category:Plant_pathogens_and _diseases
dc:subject	http://dbpedia.org/resource/Category:Sordariomycetes
foaf:page	http://en.wikipedia.org/wiki/Glomerella_tucumanensis
foaf:depiction	**http://proj.lis.ic.unicamp.br/ssea/database/shadows/ 64.pdf.xml#xpointer(dc:identifier(’erg3423’))**
dbpedia:kingdom	http://dbpedia.org/resource/Fungus
dbpedia:phylum	http://dbpedia.org/resource/Ascomycota
dbpedia:order	http://dbpedia.org/resource/Incertae_sedis
dbpedia:class	http://dbpedia.org/resource/Sordariomycetes
dbpedia:class	http://dbpedia.org/resource/Sordariomycetidae
dbpedia:abstract	”Glomerella tucumanensis is a plant pathogen.”@en
dbpedia:synonym	”Colletotrichum falcatum Went, (1893)”@en
dbpedia:synonym	”Colletotrichum metake Sacc., (1908)”@en
dbpedia:synonym	”Physalospora tucumanensis Speg., (1896)”@en
dbpedia:family	http://dbpedia.org/resource/Glomerellaceae
dbpedia:genus	http://dbpedia.org/resource/Colletotrichum

Table 5.1: Fetched RDF triples from a query to return all triples related to the “Glomerella tucumanensis” fungus

query possibilities.

5.3.3 Using Shadows to Extract Geo-knowledge

There is extensive research on extracting geographical knowledge from documents, mostly based on text analysis on documents and textual fields in databases. Basically, those papers try to find geographic references (e.g., matching place names according to a dictionary), and subsequently correlate the text with specific regions, points etc. [39, 46]. This often implies in issues of geo-referencing, token indexing and document corpus analysis algorithms (e.g., using gazeteers [36] or geographical databases [17]). Furthermore, there are problems related to the heterogeneity of formats, since most of the solutions focus on interoperable formats (e.g., HTML) or some specific format (e.g., PDF).

We decided go beyond information about species and extract geographic knowledge from non-geographic elements of documents, via linked data. Rather than analyzing the text directly to extract geo-knowledge, the basis of our strategy is to extract this knowledge from semantically annotated shadows.

Instead of restricting ourselves to geographical data, we also process data indirectly associated with geographic references (e.g., images can be connected to their meaning on the semantic web, the authors of a paper can be connected to their birthplaces, conference proceedings can be connected with where the conference took place or the address of the publishers). The middle and right parts of Figure 5.7 present an abstraction of the connection between a shadow element (addressable via URI) and its meaning in a specific data set of the Linking Open Data Project (LOD).

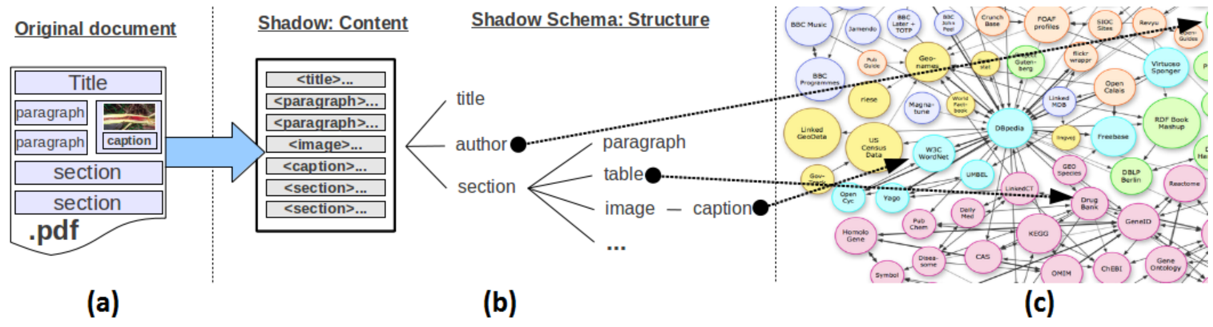
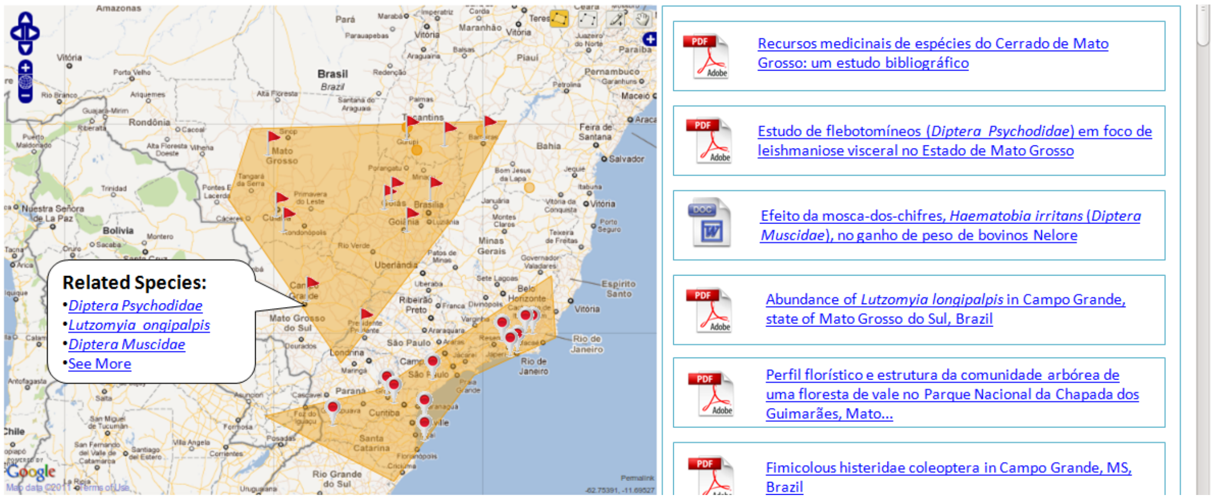


Figure 5.7: Abstraction of the relation between a shadow, the corresponding document and the link between an element and an external data set

For instance, consider a paper that contains an image of an animal. The image label identifies the species name. Using the species name, we can find its URI in the LOD, and therefore additional geo-information about that species which are dispersed over different data sets.

The paper's author can also be linked similarly, and so on. Hence, the shadow can now be used to answer queries such as "*what researchers have written papers on animals that are found within X kilometers of their work place?*", or "*group papers by geographic regions of where the species described can be found*", or "*given a document, show where mentioned species can be found*", or even "*which documents mention species that appear in a polygon P?*". Figure 5.8 shows a Web-based prototype[6] that allows a user to navigate over the geographical relationships between documents and other resources in the LOD.

[6]This prototype is not fully implemented yet



Recursos medicinais de espécies do Cerrado de Mato Grosso: um estudo bibliográfico

Estudo de flebotomíneos (*Diptera Psychodidae*) em foco de leishmaniose visceral no Estado de Mato Grosso

Efeito da mosca-dos-chifres, *Haematobia irritans* (*Diptera Muscidae*), no ganho de peso de bovinos Nelore

Abundance of *Lutzomyia longipalpis* in Campo Grande, state of Mato Grosso do Sul, Brazil

Perfil florístico e estrutura da comunidade arbórea de uma floresta de vale no Parque Nacional da Chapada dos Guimarães, Mato...

Fimicolous histeridae coleoptera in Campo Grande, MS, Brazil

Figure 5.8: Screenshot of an initial prototype for visualization of the geo-knowledge extracted from the connection between document and LOD datasets

5.4 Conclusions

This chapter presented a case study that aims to validate the SdR strategy. The case study is divided in three parts, and discusses the generation of shadows for a collection of documents, using the implementation presented in Chapter 4. Also, this chapter presented details about how shadows can be used to indirectly query the collection and to annotate its documents.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This dissertation investigates problems of handling heterogeneous documents in the Web. It proposes the *Shadow-driven Representation* (SdR), a novel strategy to represent documents independently of format, preserving the original file and handling large volumes of documents. It adopts the notion of "descriptor", borrowed from the image database literature, to represent a document's structure and content according to the interests of a groups of users.

A *shadow* is an interoperable document descriptor that summarizes key aspects and elements of a document, preserving their structural relationships. These elements (e.g., sections, tables, embedded multimedia artifacts, references) are defined by users (e.g., research groups may have different interests), and thus one document may have many shadows. Once a set of elements of interest is defined, shadows are instantiated based in this set.

Unlike other approaches in the literature that restrict document description to text, shadows consider other kinds of elements within a document, such as tables or images, thereby supporting a wide variety of operations and correlations. Though we have implemented shadows as XML documents stored in a database (our shadow base), this is just a possible materialization of the concept, which transfers document querying tasks to the DBMS.

The SdR is an approach to represent information about a document. The main advantages of this approach are:

- shadows isolate domain-relevant elements of a document from its format;
- shadows follow interoperability standards, enabling their exchange and machine consumption;

- like image descriptors, shadows concerning different information can be combined, implying in scalability;
- shadows are XML documents, and thus can be processed under IR techniques or queried via XML query languages or annotated or used for other purposes.

To validate the strategy, we implemented a working prototype to produce shadow for documents in *.doc/.docx*, *.odt*, *.pdf* formats. To show the flexibility and some possibilities of the use of shadows, we developed a case study on a specific domain – biodiversity. In the case study, shadows are used to produce semantic annotations between documents and semantic information (LOD).

6.2 Future Work

Since shadows are an interoperable "descriptor" of a document, they can be adopted as an approach to solve other problems related to documents. For instance, in the case study discussed in Chapter 5, we concentrate in problems related to documents, in two directions – query documents and their internal elements, annotate documents and link them to other resources.

There are many extensions possible for this work, for instance:

- **Defining recursive descriptors for internal elements within a document:** A document can contains several internal elements, and each element can contain its own structure and implicit information. Since a shadow acts like a descriptor of the entire document, the shadow should support the composition of specialized descriptors – such as image descriptors, descriptors that describe a table etc.
- **Clustering documents according to their hierarchical structure:** Internally, a shadow contains a representation of the internal elements of a document. This representation preserves the hierarchical relationship between the elements. Thus, the shadow contains a "virtual" representation of the hierarchical relationship of the elements. This virtual representation of the structure can be used as a criterion for document clusterization and classification – e.g., all PhD theses, even in different languages may have a common structure.
- **Representing elements of documents as RDF triples:** In the implementation presented in this dissertation, shadows are serialized as XML documents. Another possible strategy to construct and serialize shadows is the construction of a graph of RDF triples that represents both structure and instances of elements within a

document. With this strategy, semantic web systems could transparently process the content and/or the structure of documents.

- **Defining the shadow schema in an ontology:** In the implementation discussed in this dissertation, a shadow schema drives the process of extraction of elements and instantiation of shadows. The current implementation has the extractable types of elements hardwired in the code. Another approach to drive the extraction and production process is the use of ontologies. Basically, an "ontology schema" should contain specialized directives in order to recognize specific types of elements in different document formats, according to user needs. This ontology could drive the extractor, decoupling the code from the element definition itself.

- **Using shadows for document versioning and derivation discovery:** Many document production tools are not conceived to produce files with explicit structure and interoperable formats, strongly coupling the content to the file structure and software representation. This hampers processes like versioning and discovery of correlations between different formats. Since a shadow isolates the content from the format, shadows can be used for solving these problems.

- **Using shadows as input for NLP algorithms:** There are several papers that process documents with NLP techniques with different purposes,focusing textual features. Usually, systems that go beyond textual features concentrate in specific formats, like HTML. As an alternative strategy, NLP algorithms could use shadows as input, also considering information about the structure of the document – for instance, the label of a section can have more relevance on the processing than a caption of an image.

- **Building drivers to store shadow in other DBMS:** DDEx extracts elements from documents, requiring different drivers for specific formats. An extension to the architecture is to develop drivers that will store shadow in distinct DBMS, which will allow a variety of shadow access and query mechanisms.

Bibliography

[1] Y. Alemu, J.-b. Koh, M. Ikram, and D.-K. Kim. Image Retrieval in Multimedia Databases: A Survey. *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 681–689, Sept. 2009.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer Berlin / Heidelberg, 2007.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):28–37, May 2001.

[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[5] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165, 2009.

[6] P. Cano. Automatic sound annotation. In *In IEEE workshop on Machine Learning for Signal Processing*, pages 391–400, 2004.

[7] P. Chan, H. Yu, W. Ng, and D. Yeung. A novel method to reduce redundancy in adaptive threshold clustering key frame extraction systems. In *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, volume 4, pages 1637 –1642, july 2011.

[8] M. V. Cundiff. An introduction to the Metadata Encoding and Transmission Standard (METS). *Library Hi Tech*, 22(1):52–64, 2004.

[9] H. Cunningham. GATE, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

[10] R. da Silva Torres and A. Falcão. Content-based image retrieval: Theory and applications. *Revista de Informática Teórica e Aplicada*, 2(13):161–185, 2006.

[11] DCMI. Dublin core metadata initiative, Aug. 2010. `http://dublincore.org/metadata-basics/`, accessed on 01/2011.

[12] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language for xml. *Computer Networks*, 31(11–16):1155 – 1169, 1999.

[13] E. Duval, W. Hodgins, S. Sutton, and S. L. Weibel. Metadata Principles and Practicalities. *D-Lib Magazine*, 8(4):1–10, Apr. 2002.

[14] J. Euzenat. Eight questions about semantic web annotations. *IEEE Intelligent S.*, 17(2):55–62, 2002.

[15] R. Freitas and R. Torres. An ontology-based tool for image retrieval and semi-automatic annotation (in portuguese). 2005.

[16] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides. *Design Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1995.

[17] L. C. Gomes and C. B. Medeiros. Ecologically-aware queries for biodiversity research. In *Proceedings GeoInfo - Brazilian Geoinformatics Symposium*. INPE - SBC, 2007.

[18] J. Greenberg. Understanding Metadata and Metadata Schemes. *Cataloging & Classification Quarterly*, 40(3):17–36, Sept. 2005.

[19] C. Grün, S. Gath, A. Holupirek, and M. H. Scholl. Xquery full text implementation in basex. In *Proceedings of the 6th International XML Database Symposium on Database and XML Technologies*, XSym '09, pages 114–128, Berlin, Heidelberg, 2009. Springer-Verlag.

[20] C. Grün, A. Holupirek, and M. H. Scholl. Visually exploring and querying xml with basex. In *BTW*, pages 629–632, 2007.

[21] T. Grust, M. Mayr, and J. Rittinger. Let SQL Drive the XQuery Workhorse. In *Proc. EDBT*, pages 147–158, 2010.

[22] X. Han, L. Sun, and J. Zhao. Collective Entity Linking in Web Text: A Graph-Based Method. In *Proc SIGIR*, pages 765–774, 2011.

[23] B. Haslhofer. A survey of techniques for achieving metadata interoperability. *ACM Computing Surveys (CSUR)*, 42(2):1–37, Feb. 2010.

[24] T. Hey, S. Tansley, and K. M. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[25] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank. A survey on visual content-based video indexing and retrieval. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(6):797 –819, nov. 2011.

[26] E. Ioannou, W. Nejdl, C. NiederZee, and Y. Velegrakis. OntheFly Entity Aware Query Processing in the Presence of Linkage. In *Proc. VLDB*, pages 279–287, 2010.

[27] J. Kahan. Annotea: an open RDF infrastructure for shared Web annotations. *Computer Networks*, 39(5):589–608, Aug. 2002.

[28] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49 – 79, 2004.

[29] M. Koivunen, R. Swick, and E. Prud'hommeaux. Annotea shared bookmarks. In *Proc. of the KCAP 2003 workshop on knowledge markup and semantic annotation, http://www. w3. org/2001/Annotea/Papers/KCAP03/annoteabm. html*, pages 25–26. Citeseer, 2003.

[30] H. Lejsek, F. H. Ásmundsson, B. T. Jónsson, and L. Amsaleg. Scalability of local image descriptors: a comparative study. In *MULTIMEDIA '06*, pages 589–598, NY, USA, 2006. ACM.

[31] M. Lesaffre, K. Tanghe, G. Martens, D. Moelants, M. Leman, B. D. Baets, H. D. Meyer, and J.-P. Martens. The mami query-by-voice experiment: Collecting and annotating vocal queries for music information retrieval. In *In: Proceedings of the International Conference on Music Information Retrieval*, pages 26–30, 2003.

[32] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262 – 282, 2007.

[33] C. Macário, S. Sousa, and C. B. Medeiros. Annotating Geospatial Data based on its Semantics. In *17th ACM SIGSPATIAL Conference*, pages 81–90, 2009.

[34] S. Marinai, B. Miotti, and G. Soda. Digital Libraries and Document Image Retrieval Techniques: A Survey. *Learning Structure and Schemas from Documents*, pages 181–204, 2011.

[35] M. S. Mota, J. S. C. Longo, D. C. Cugler, and C. B. Medeiros. Using linked data to extract geo-knowledge. In *XII Brazilian Symposium on GeoInformatics - GeoInfo*, November 2011.

[36] D. Nadeau, P. Turney, and S. Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Advances in Artificial Intelligence*, volume 4013 of *Lecture Notes in Computer Science*, pages 266–277. 2006.

[37] A. Nandy, M. Harle, and S. Basak. Mathematical descriptors of DNA sequences: development and applications. *Arkivoc*, 9(ix):211–238, 2006.

[38] NISO. *Understanding metadata*. National Information Standards Organization, 2004. ISBN 1-880124-62-9.

[39] R. Odon de Alencar, C. A. Davis, Jr., and M. A. Gonçalves. Geographical classification of documents using evidence from wikipedia. In *Proceedings of the 6th Workshop on Geographic Information Retrieval*, GIR '10, pages 12:1–12:8. ACM, 2010.

[40] E. Oren, K. H. Moller, S. Scerri, S. Handschuh, and M. Sintek. What are semantic annotations?? 2006. Technical report, DERI Galway.

[41] N. Patil, D. Toshniwal, and K. Garg. Species identification based on approximate matching. In *Proceedings of the Fourth Annual ACM Bangalore Conference*, COMPUTE '11, pages 30:1–30:4, New York, NY, USA, 2011. ACM.

[42] L. Ren, Z. Qu, W. Niu, C. Niu, and Y. Cao. Key frame extraction based on information entropy and edge matching rate. In *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, volume 3, pages V3–91 –V3–94, may 2010.

[43] A. Santanchè and C. B. Medeiros. A Component Model and Infrastructure for a Fluid Web. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):324–341, February 2007.

[44] A. Santanchè, M. Mota, D. Costa, N. Oliveira, and C. O. Dalforno. Componere: component-based in web authoring. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, WebMedia '09, pages 12:1–12:8, New York, NY, USA, 2009. ACM.

[45] N. C. Simões, N. J. Leite, and B. Marcotegui. Automatic key-frame extraction from broadcast soccer videos. In *VISAPP (2)*, pages 216–223, 2009.

[46] J. Strötgen, M. Gertz, and P. Popov. Extraction and exploration of spatio-temporal information in documents. In *Proceedings of the 6th Workshop on Geographic Information Retrieval*, GIR '10, pages 16:1–16:8, New York, NY, USA, 2010. ACM.

[47] C.-W. Su, H.-Y. Liao, H.-R. Tyan, C.-W. Lin, D.-Y. Chen, and K.-C. Fan. Motion flow-based video retrieval. *Multimedia, IEEE Transactions on*, 9(6):1193 –1201, oct. 2007.

[48] A. Termehchy and M. Winslett. Keyword Search for Data-Centric XML Collections with Long Text Fields. In *Proc. EDBT*, 2010.

[49] F. Valio, H. Pedrini, and N. Leite. Fast rotation-invariant video caption detection based on visual rhythm. In C. San Martin and S.-W. Kim, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 7042 of *Lecture Notes in Computer Science*, pages 157–164. Springer Berlin / Heidelberg, 2011.

[50] J. van Ossenbruggen, F. N., and L. Hardman. That Obscure Object of Desire: Multimedia Metadata on the Web, Part 1. *IEEE MultiMedia*, 11(4):38–48, 2004.

[51] L. M. N. Walsh. *DocBook: The Definitive Guide.* O'Reilly & Associates, 1999.

[52] N. Walsh and L. Muellner. *DocBook: The Definitive Guide.* O'Reilly, 1999.

[53] D. Wang and T. Li. Document Update Summarization Using Incremental Hierarchical Clustering. In *Proc. CIKM*, pages 279–287, 2010.

[54] L. Weng, Z. Li, R. Cai, Y. Zhang, Y. Zhou, L. T. Yang, and L. Zhang. Query by document via a decomposition-based two-level retrieval approach. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 505–514, New York, NY, USA, 2011. ACM.

[55] E. Wilde and D. Lowe. *Xpath, Xlink, Xpointer, and Xml: A Practical Guide to Web Hyperlinking and Transclusion.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[56] D. Wimalasuriya and D. Dou. Components for Information Extraction: Ontology-Based Information Extractors and Generic Platforms. In *Proc. CIKM*, pages 9–18, 2010.

[57] S. H. Yeganeh, O. Hassanzadeh, and R. J. Miller. Linking semistructured data on the web. In *Proceedings of the 14th International Workshop on the Web and Databases*, 2011.

[58] L. Zhang, Y. Zhang, and Q. Xing. Filtering semi-structured documents based on faceted feedback. In *Proceedings of the 34th international ACM SIGIR conference*

on Research and development in Information, SIGIR '11, pages 645–654, New York, NY, USA, 2011. ACM.