Discovering Interesting Information with Advances in Web Technology

Article *in* ACM SIGKDD Explorations Newsletter · December 2012

DOI: 10.1145/2481244.2481255

Richi Nayak
Queensland University of Technology
206 PUBLICATIONS **2,468** CITATIONS

Pierre Senellart
MINES ParisTech
90 PUBLICATIONS **1,932** CITATIONS

Aparna S. Varde
Montclair State University
172 PUBLICATIONS **1,184** CITATIONS

Discovering Interesting Information with Advances in Web Technology

Richi Nayak[1], Pierre Senellart[2], Fabian M. Suchanek[3], and Aparna S. Varde[4]
1. School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, Australia
2. Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI, Paris, France
3. Ontologies Group, Max Planck Institute for Informatics, Saarbrücken, Germany
4. Department of Computer Science, Montclair State University, NJ, USA

r.nayak@qut.edu.au, pierre@senellart.com, fabian@suchanek.name, vardea@montclair.edu

ABSTRACT

The Web is a steadily evolving resource comprising much more than mere HTML pages. With its ever-growing data sources in a variety of formats, it provides great potential for knowledge discovery. In this article, we shed light on some interesting phenomena of the Web: the deep Web, which surfaces database records as Web pages; the Semantic Web, which defines meaningful data exchange formats; XML, which has established itself as a lingua franca for Web data exchange; and domain-specific markup languages, which are designed based on XML syntax with the goal of preserving semantics in targeted domains. We detail these four developments in Web technology, and explain how they can be used for data mining. Our goal is to show that all these areas can be as useful for knowledge discovery as the HTML-based part of the Web.

1. INTRODUCTION

Recent years have seen several new developments in the realm of the World Wide Web. Many of these developments are related to improving responses to user queries and discovering knowledge that is useful for user applications. In this survey article we focus on some major developments, namely, the deep Web, the Semantic Web, XML (eXtensible Markup Language) and DSMLs (domain-specific markup languages). We discuss how each of these four developments assist in discovering interesting and useful information from the Web.

The deep Web is the part of the Web that is not stored as HTML pages, but that is generated on demand from databases [16; 46; 89]. The Semantic Web [107] is the extension of the Web that makes data available in semantic data formats. It incorporates standards such as RDFS [109] which serve to define and interlink ontologies. XML has become a widely accepted standard in information exchange because of its self-explanatory tags that allow the storage of information in semi-structured format [114]. This has led to research in the field of mining the data represented in XML [33; 77]. It has also encouraged the development of DSMLs [44; 99]. These are designed based on the XML syntax and are tailored to specific domains, thus catering to the needs of targeted user bodies.

This article is motivated by the potential of utilizing these four developments to improve the user experience of the Web. As a motivating example, consider the works of the physicist Albert Einstein. Some data pertaining to his work cannot be found on the surface Web. Details about co-authors, e.g., can be found only by posing queries to discover knowledge from the deep Web. Trivia about Einstein including his birth details, Nobel Prize information, and so forth can be better obtained with the help of the Semantic Web. It uses standards such as RDFS, which add meaning to the data. XML mining techniques, taking advantage of underlying semantic and structural information, can be used to group and search Einstein's prominent papers on pertinent topics and make comparisons between his works and those of his contemporaries. DSMLs can help to code the actual content of his formulae and present them in suitable formats over the Web. This way, they are not treated merely as pictures, but instead can be queried and used for knowledge discovery in conjunction with mining paradigms. This example shows how each of these four developments can be helpful for discovering various forms of interesting information across the Web. This article provides a review of the deep Web, Semantic Web, XML and DSMLs, focusing on essential concepts, and explaining how these developments enhance the discovery of useful information.

The rest of this survey article is organized as follows. Section 2 describes how knowledge can be discovered from the deep Web, covering both extensional and intensional approaches and related topics [83; 23; 113; 22]. Section 3 explains how the Semantic Web can help retrieve meaningful information, using RDF, RDFS, OWL, and SPARQL [108; 109; 4; 111; 92]. Section 4 focuses on knowledge discovery from the data stored using XML. We provide insights on the benefits and challenges that XML mining research brings and explain how the XML data can be modeled and mined with a focus on clustering and frequent-pattern mining [2; 53; 62]. Section 5 focuses on domain-specific markup languages. We explain their usefulness in storing and retrieving data within a given domain, and also emphasizing their significance in data mining and knowledge discovery [30; 102; 115] along with suitable examples. Section 6 provides a general outlook and discusses some challenges and open issues in all these areas. Finally, Section 7 presents the summary and conclusions.

2. THE DEEP WEB FOR DISCOVERING HIDDEN KNOWLEDGE

2.1 The Deep Web

The whole content of the Web cannot be reached by just following hyperlinks. Web-accessible databases such as weather information services or library catalogs usually need to be queried by filling in fields of a HTML form. Even when the information is accessible by following links, it can be more effective to find it through a (possibly elaborate) Web form query. The terms *hidden Web* [83], *deep Web* [16], *invisible Web* [84] have been used in the literature to describe more or less the same thing: the part of the Web which is not accessible through hyperlinks, and that is typically constructed from databases. Both terms *hidden* and *invisible* insist on the inaccessibility of this information to search engines, while *deep* insists on the fact that the information lies in databases behind forms, and requires *deeper* crawls than the usual *surface* crawls of search engines.

A 2000 study from the company BrightPlanet [16] has had much impact on the development of research about the deep Web. This study uses correlation analysis between search results of different search engines to estimate the size of the deep Web; they found that it contains between 43,000 and 96,000 Web databases, with around 500 times more content than the surface Web. In other words, the largest part of the content present on the Web is not exploitable by classical search engines. Although this kind of estimation is by nature quite imprecise, other more recent works [46] confirm this order of magnitude with another estimation approach and come up with a number of around 400,000 databases (this takes into account the growth of the Web between the times of the two studies). Moreover, even directories that are specializing in listings databases on the Web—a large list of these is given in [84]—have poor coverage of the services of the deep Web (15% at best, cf. [46]). This is a clear motivation for designing systems that discover, understand and integrate deep-Web services.

Though these numbers may be surprising at first, it is easy to see that there is a lot of content on the deep Web: *Yellow pages* services and other forms of directories (it is likely that the phone numbers of a given person are available in deep Web databases, such as a phone company directory, or an internal directory of some institution, even though they might not appear on the surface Web), library catalogs, weather forecast services, geolocalization services, administrative sources, such as the database of the United States census bureau, etc. Moreover, there are cases when it makes sense to access content of the surface Web through services of the deep Web, to use them in a more semantic way. This is for instance the case with e-commerce Web sites, which present their catalogs on the surface Web in order to be indexed by search engines, but allow for fine-tuned querying when accessed through advanced search forms. This is also the case if you want to retrieve, say, the names of all co-authors of Albert Einstein; though all his articles are probably available on the surface Web, it is much more effective to ask the corresponding query to a scientific publication database service like Google Scholar.

The question is, how to benefit from this information. How to discover it, to index it, and to be able to use it when answering a user's query? We focus especially on automatic approaches for knowledge discovery over the deep Web, with as little human interaction and supervision as possible.

2.2 Mining the Deep Web

We define here some of the important concepts related to the deep Web. A *response page* is a page that is generated in response to some request, in most cases from data stored in a database. A *query form* is a Web page with an HTML form that, when submitted, triggers the creation of a response page. A query form consists of multiple *fields*, where each field can be filled with either a value from a predefined list or with free text. A *Web service*, as it is defined by the World Wide Web consortium, is ”a software system designed to support interoperable machine-to-machine interaction over a network” [112]. Web services are an arguably rare form of content of the deep Web, and are typically formally described using WSDL, the Web Service Description Language, a W3C standard. We assume we are mostly interested in deep Web content pertaining to some *domain of interest* (e.g., research publications), described by a set of *domain concepts* (which can just be strings like “author” or “publication”, or something more elaborate, e.g., an element of an ontology, see Section 3).

A survey of a number of systems that deal with services of the deep Web can be found in [84]. Most current approaches can be roughly classified into extensional strategies (retrieving information from the deep Web and storing it locally to process it) and intensional strategies (analyzing services to understand their structure, store this description, and use it to forward users' queries to the services). We discuss next both approaches, before going into more detail of the individual tasks that such systems need to perform.

Extensional Approach. In the extensional approach, shown in Figure 1, the first step is to discover sources of the deep Web of interest, say, from a given domain. The entry point to these sources will typically be an “advanced search” HTML form. From then, the extensional approach consists in trying to submit this query form with some sample words, in order to siphon the entire content of the Web site, resulting in a large number of result pages. We probably want to use some kind of bootstrapping, i.e., to start with a small set of words to enter in the query form, that might come from some domain knowledge, and then discover new words from the result pages to query further the form. Once a sufficient portion of the content of the Web site has been siphoned, result pages can be indexed and queried as if there were pages of the surface Web.

One of the first practical works on the deep Web [83] follows the extensional strategy: HiWe is a general system for representing forms and mapping their fields to concepts (relying heavily on human annotation), so as to retrieve result pages that are cached in a local index. This approach has several advantages, including a relative domain independence, effectiveness, and efficiency. By storing deep Web pages like other regular Web content, this approach also allows searching indiscriminately deep Web and surface Web data. However, it also poses a considerable load on the source because it siphons the source entirely. Google has been experimenting with this approach [67] (the authors call it *surfacing*).

The main challenges raised by the extensional approach are the following. First, a knowledge discovery system from the deep Web needs to discover services of interest. Next, it

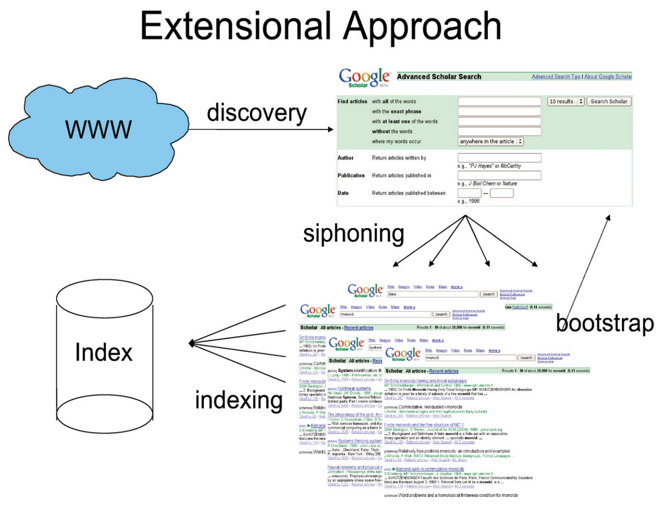

Figure 1: The Extensional Approach

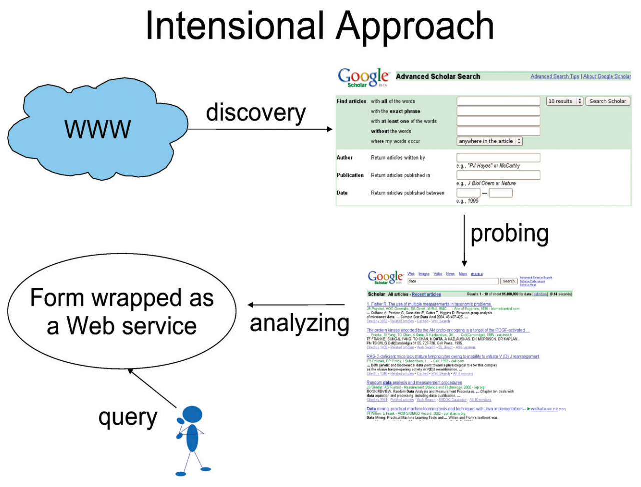

Figure 2: The Intensional Approach

needs to select relevant data to submit forms with. Data found in result pages can be used to bootstrap this siphoning process. And we need the whole process to cover some large enough portion of the database. Note that the extensional approach is not always feasible, because it either imposes a high load in terms of bandwidth and computation time on Web servers, or requires a very long time to siphon an entire database (a whole month would be required to retrieve ten million Web pages without asking more than one page every ten seconds). If such an approach were to become commonplace, it is likely that new crawling ethics and guidelines would have to be agreed upon by Web content providers and search engines.

Intensional Approach. The intensional approach, shown in Figure 2, starts as the extensional one by discovering services of interest. The result pages are also generated as in the extensional approach, but much more parsimoniously: the goal here is not to siphon the entire database, but to retrieve enough pages (maybe only one, or a handful of them) to have enough data for analyzing the structure of these pages, i.e., analyzing how they are formatted. If a system is able to analyze both the structure of input forms (to know which fields represent which concepts), and the structure of output pages, then it is essentially able to deal with a form made for human as with a Web service made for machines. Therefore when a user issues a query relevant to the domain of the form, this query is translated, rewritten, using the querying capabilities of the form, and the relevant results extracted from the result pages are presented to the user.

Extensive research about the intensional indexing of the deep Web was led by a group at University of Illinois at Urbana-Champaign, with some external collaborations [23; 113; 22]. Two systems that have been developed in this context are MetaQuerier and WISE-Integrator. The aim is to discover and integrate sources of the deep Web for querying them in a uniform way. There is a strong focus in these works on schema matching between different sources. MetaQuerier [23] uses a *holistic* approach, that is, matching a number of different sources at once, rather than pair-wise matching. In particular, the global schema is typically not predefined but derived from clustering of the fields from different sources. This type of work essentially focuses on the analysis of the syntax of the form and the discovery of concepts in its different fields (some recent works [113; 22] also deal with the analysis of the result pages).

An intensional approach is obviously more ambitious than the extensional one. The main issues here are first, as before, discovering services of the deep Web. Then, we now need to really understand the structure (and, hopefully, semantics) of both a Web form and a result page. In the latter case, we actually want to build, automatically, a wrapper for this kind of result page. Understanding what inputs a service expects (form analysis) and what outputs it produces (result page analysis) may actually not be sufficient. The problem is that a machine cannot easily know the semantics of a service. For example, assume that a service takes as input a person name and yields as output another person name. Is it a genealogical service that gives ancestors? A human resource service giving supervisors? Therefore, theoretically, a true semantic analysis of the relations between the input and output of a service should be performed, but this is a very deep and complex problem. In some domains of interests (e.g., scientific publication databases), there are actually not many semantic ambiguities: a name is usually (but not always!) the author of a paper and a year the publication date. Finally, an advantage of an intensional approach is that it does not impose any high load on Web servers, since we are only interested in generating result pages that help us analyzing the structure of the service, and is as such much more scalable than the extensional one. Then processing a user's query can result in further queries to the service.

Note that, in contrast with the extensional approach, the outcome of the intensional one is very different from regular Web content: it is a semantic description of services, that can be queried when needed for answering a user's query. As such, it relates more to semantically described data in the spirit of the Semantic Web than to regular Web pages.

2.3 Deep Web Mining: Components

We discuss next the most important steps of both the extensional and intensional approaches.

Discovery of Services. The first task needed by a knowledge discovery system for the deep Web is to find deep Web sources of interest. It basically includes crawling the Web and selecting some HTML forms (deep Web sources might also be accessible through SOAP Web services, but this is rare). The problem is, not all forms are entry points to a deep Web database. We certainly do not want to submit hotel reservation or mailing list subscription forms, for instance. It is also probably not interesting to see a simple keyword search interface (say, Google search) over a Web site or a collection of Web sites as a deep Web source, since it will only retrieve pages from the surface Web. When we crawl the Web to discover services, we have to detect services with side effects to exclude them and it is not easy to do so. We shall have to rely on heuristics such as avoiding services requiring the input of an email-address, a credit-card number, or a password (unless this service has been explicitly declared by a user of the system). As already mentioned, we are only interested in services relevant to some domain of interest. If, for instance, we crawl the Web to discover services, we must "focus" the crawl to the domain of interest; this kind of *focused crawling* of the Web has been studied in [19]. An interesting approach to the specific problem of focused crawling for discovering forms is presented in [9]. The authors combine classical focused-crawling techniques, with a page classifier for the domain of interest, with two sources of feedback that help controlling it: a form classifier that checks whether a form is of interest, and a link classifier that considers features of the history of the links followed to get to the current page.

Form Analysis. Although some of the sources of the deep Web are Web services described in WSDL, these are minority. Most of the services that can be found on the deep Web are accessible through an HTML form interface, and their results are shown as HTML result pages. In order to both understand and index these services, we need to understand the structure of form interfaces, that is to map each field with some concept of the domain of interest. We assume here that fields can be queried independently from each other, which is a common (but not always valid) assumption. A number of techniques have been proposed [83; 55; 18; 90], mostly based on the idea of *probing*. We present briefly the idea of [90]. This is a two-phase process, the first phase being quite straightforward. First, we build a textual context for each field (`label` tag associated to the field, `id` or `name` attributes, all surrounding text). Standard text preprocessing techniques such as stop-word removal, stemming, etc are applied. The resulting bag of tokens can then be matched with domain concepts. This straightforward process gives us candidate annotations, which are usually of rather good quality (very high recall, high precision), but we do not stop here. For each candidate annotation with concept c that the system has just obtained, the following steps are applied. First, the field is probed (and the form submitted) with some nonsense word, such as the one obtained when one types randomly on a keyboard. Hopefully, the result page should be an error page stating that there were no results to the query. Then, the field is probed with words that appear in instances of concept c (e.g., last names for a field expecting a person). The pages obtained by probing with an instance word are then compared to the error page. A structural clustering (for instance based on the DOM tree of the page) can be obtained for that purpose. If enough result pages that are not similar to the error page are obtained, the result can be confirmed. Experiments show this confirmation phase improves precision vastly without hurting recall. In the extensional case, we actually want to submit the form repetitively, with a huge number of instance words, in order to generate as many result pages as possible, and hopefully to cover the whole content of the deep Web database. We can bootstrap this with just a few example words, if we use words found in the result pages (and especially, frequent words) to submit the form.

Result Page Analysis. In the intensional case, once a form is understood, the structure of result pages needs to be understood as well. Typically, result pages present a set of records with fields, with the same presentation for a given deep Web database. But this presentation is unknown. It can be a HTML list, a table or a set of paragraphs separated by whitespace. The structure of fields inside this set of records can also be quite diversified. The goal is to be able to build wrappers for a given kind of result pages, in a fully automatic, unsupervised, way. Unsupervised wrapper induction is a very complex problem. We are just going to give a sketch of what can be done. We start with example result pages, that are unlabeled. The main idea is to use the repetitive structure of these pages to infer a wrapper for all pages sharing the same structure. Thus, if we detect a table structure with various number of lines, we might be able to infer automatically that the records are contained in the rows of such tables. Systems like RoadRunner [27] or ExAlg [5] perform this unsupervised wrapper induction by inferring some regular expression that describes best (and most concisely) the repetitive structure (e.g., DOM tree) of a page. This is unfortunately not enough. We do not want only to know where the data is, but we also want to label it. Labeled unsupervised wrapper induction is even harder. A promising idea, developed in [90], is to use a structural wrapper induction technique in conjunction with a technique that relies on the content itself. We can use the domain knowledge to first annotate the original documents (named entity recognizers can recognize some tokens as a person name, as a date, etc.), and then use the structure of the document to generalize the annotation. This is probably akin to what humans perform when they are presented with a result page: first, they recognize some known tokens (person names, dates, things looking like titles), and then they infer that the person names are thus located in this particular place on the page, e.g., as hyperlinks inside the second column of some table.

Scalability. As Web sites can be processed independently of one another, and each operation (form classification, form and result page analysis), however costly, only applies to a small number of pages per site, the intensional approach is fairly scalable – the number of deep Web sources processed increases linearly with added computing and network resources. On the other hand, the extensional approach, by siphoning entire Web databases, is sometimes not usable on large Web sites.

On the basis of this discussion about the background of the deep Web and its mining approaches, we can see that the deep Web does indeed present considerable potential for knowledge discovery.

3. THE SEMANTIC WEB TO RETRIEVE MEANINGFUL INFORMATION

Recent years have seen a steady increase of publicly available data. This includes, for example, government census data, life science data, scientific publications, user generated content on social networks or natural language Web pages. These data are scattered all over the globe, residing on different servers in a multitude of formats. Some data are available as HTML or XML files, others as deep Web sites and again others as downloadable data files. This entails that an application cannot easily bridge data from different sources. For example, a library system in Germany cannot easily access books stored in a French library system, an airline booking system cannot interact with an airport shuttle service, and smart phone is currently not smart enough to figure out the opening hours of a restaurant.

These barriers amongst data sources gave rise to the idea of the Semantic Web. The Semantic Web is an evolving extension of the Web, in which data is made available in standardized, machine-readable semantic formats. On a more abstract level, the Semantic Web project aims at defining the semantics of data and services, at facilitating the sharing of information across different servers and systems and, ultimately, at allowing machines to understand and answer semantic knowledge requests. The project is driven by the World Wide Web Consortium, in collaboration with academic and industrial partners. Currently, the Semantic Web comprises several dozen knowledge bases with billions of units of knowledge [12]. These include general semantic knowledge bases about the entities of this world (such as Albert Einstein or the Theory of Relativity) as well as domain-specific data sources, such as movie collections, bibliographies, or geographic databases. All of this data is available under permissive licenses, and in one standardized format, which makes it particularly interesting for data mining. This section will introduce the building blocks of the Semantic Web and then explain how the semantic data can be mined.

3.1 Building Blocks of the Semantic Web

URIs and Namespaces. For the Semantic Web, the basic units of knowledge are *entities*. An entity is an abstract or concrete thing, whether it exists or not. For example, Albert Einstein, the Theory of Relativity, and the city of Ulm are all entities. Knowledge bases (KBs) define properties of entities. For example, a biographical KB may define the birth date and the profession of Albert Einstein.

The KBs refer to the entities by identifiers. For instance, a KB might refer to Albert Einstein by the identifier *AlbertEinstein*. In order to bridge knowledge across different KBs, we have to make sure that no two sources use the same identifier for different entities. This is the purpose of *Uniform Resource Identifiers* (URIs). A URI can be understood as a generalized form of an Internet address. Unlike Internet addresses, URIs do not have to be accessible on the Internet. For example, `http://biographies.com/AlbertEinstein` is a valid URI, even if it cannot be accessed in a browser. Since Internet domains are allocated globally and since domain owners can assign sub-domains and paths at their own discretion, a URI used by one KB can be guaranteed to be different from the URIs used by other KBs. This way, the domains form disjoint namespaces for URIs. Technically, a URI is a string that follows a certain syntax specification [78]. Since URIs are quite verbose, it is common to use *namespace prefixes*. A namespace prefix is an abbreviation for the prefix of a URI. For example, throughout this article, we will use the string `b` to abbreviate the URI prefix `http://biographies.com/`. Then, instead of writing `http://biographies.com/AlbertEinstein`, we can equivalently write `b:AlbertEinstein`. Such an abbreviated URI is called a *qname*. Qnames are a purely notational simplification of URIs.

RDF. In order to make knowledge from different KBs interoperable, the knowledge has to be represented in one unified format. This is the purpose of the *Resource Description Format* (RDF) [108]. RDF is a knowledge representation model that closely resembles the Entity Relationship model. Each piece of information is represented as a *statement* in RDF, i.e., as a triple of three URIs: a subject, a predicate and an object. The statement says that the subject and the object stand in the relation given by the predicate. For example, the following statement says that Albert Einstein received the Nobel Prize:

```
b:AlbertEinstein  b:received  b:NobelPrize
```

Relationships (such as `b:received`) are also entities. They are identified by URIs and can also be the subject or the object of a statement. By using URIs, a statement can easily span entities described in different KBs. For example, assume that some other KB in the namespace `http://nobelpriset.sv/` defines the entity `NobelPrize`. The following statement re-uses the identifier from this KB:

```
b:AlbertEinstein  b:received
                  http://nobelpriset.sv/NobelPrize
```

RDF predicates are always binary relationships. Sometimes, relationships with higher arity are necessary. For example, to state that Albert Einstein received the Nobel Prize for the discovery of the photoelectric effect, we would need a ternary relationship. In RDF, the standard way to represent these relationships is to introduce a new entity, an *event entity*. This event entity is then linked by binary relations to each of the arguments. In the example, we could introduce an event entity `b:e42`, and the binary relations `b:laureate`, `b:prize`, and `b:discovery`). Then, the sample fact can be represented as:

```
b:e42  b:laureate   b:AlbertEinstein
b:e42  b:prize      b:NobelPrize
b:e42  b:discovery  b:PhotoelectricEffect
```

Standard Vocabulary. By RDF statements, a KB can define and describe entities, including relationships. Other KBs can refer to these entities by URIs. In the ideal case, all KBs would use the same URI for the same entity. In reality, many KBs define their own URI for an entity. However, there are a number of standard KBs that provide a

Table 1: Common Namespace Prefixes

rdf:	The basic RDF vocabulary (see Section 3.1) `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
rdfs:	RDF Schema vocabulary (see Section 3.1) `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
owl:	Web Ontology Language (see Section 3.1) `http://www.w3.org/2002/07/owl#`
xsd:	Data types `http://www.w3.org/2001/XMLSchema#`
dc:	Dublin Core (predicates for describing documents) `http://purl.org/dc/elements/1.1/`
dbp:	The DBpedia ontology (real-world entities) `http://dbpedia.org/resource/`
yago:	The YAGO ontology (real-world entities) `http://yago-knowledge.org/resource/`
foaf:	Friend Of A Friend (relationships between people) `http://xmlns.com/foaf/0.1/`
cc:	Creative Commons (types of licences) `http://creativecommons.org/ns#`

standard vocabulary of entities that is often re-used across KBs. Table 1 lists some of them with their usual namespace prefixes.

RDFS. Entities with similar properties are grouped in a *class*. For example, people involved in physics are grouped in a class `physicist`. In RDF, a class is referred to by a URI and it can appear in statements. For example, to state that Einstein is a physicist, one would write:

```
b:AlbertEinstein  rdf:type  b:physicist
```

We are using the `rdf` namespace prefix here to reference the relationship `type` that is defined by the RDF standard vocabulary (see Table 1). The RDF standard vocabulary defines a very limited set of entities. It is extended by the vocabulary of the *Resource Description Framework Schema* (RDFS) [109]. RDFS defines relationships such as `subClassOf` and `domain`, which can be used to state that a class is a sub-class of another class or that a predicate has a certain domain, respectively:

```
b:physicist  rdfs:subClassOf  b:scientist
b:received   rdfs:domain      b:person
```

The hierarchy of classes with their super-classes is called a *taxonomy*. RDFS also defines a number of logical entailment rules, which define logical consequences of RDFS statements.

OWL and Ontologies. To specify consistency of KBs, we can use the *Web Ontology Language OWL* [4]. Technically, OWL is an additional KB that defines a vocabulary of entities and relationships. To say, e.g., that `b:bornInYear` is a many-to-one property, we can use one of the classes defined in OWL:

```
b:bornInYear  rdf:type  owl:FunctionalProperty
```

OWL also defines the vocabulary for set operations on classes, restrictions on properties and equivalence of classes. For example, we can state that the classes of meat and fruit are disjoint:

```
b:meat  owl:disjointWith  b:fruit
```

The OWL vocabulary comes with a definition of entailment: it specifies which statements logically entail which other statements. It also comes with a definition of inconsistency, i.e., which statements are contradictory. For example, a consistent KB cannot contain statements using a functional predicate with more than one object for a given subject.

Table 2: Some RDF knowledge bases

Name	URL	Size
Freebase	`http://www.freebase.com` (community collaboration)	100m
DBpedia [6]	`http://www.dbpedia.org` (from Wikipedia, focus on interlinking)	1b
YAGO [92]	`http://yago-knowledge.org/resource/` (from Wikipedia, focus on accuracy)	120m
Bio2RDF	`http://www.bio2rdf.org` (from biological data sources)	2.4b
MusicBrainz	`http://www.musicbrainz.org` (Artists, Songs, Albums)	23k
Geonames	`http://www.geonames.org` (Countries, Cities, Capitals)	93m
DBLP	`http://dblp.l3s.de/d2r/` (Papers, Authors, Citations)	28m
US Census	`www.rdfabout.com/demo/census` (Population statistics)	1b

There are dozens of large-scale KBs on the Semantic Web. Table 2 lists some of the largest ones.[1] In the Semantic Web world, the word *ontology* is often used synonymously with *knowledge base*. In a stricter sense of the word, *ontology* refers to the *T-Box* of a KB. The T-Box of a KB is the definition of classes, relations, and constraints, i.e., roughly the statements with relations from the namespaces of RDFS and OWL.

3.2 Mining the Semantic Web

RDF Access. The standard way of exchanging RDF data is by encoding the statements in XML [110]. Since the XML syntax for RDF is quite verbose, RDF data is often just expressed as space-separated triples of subject, predicate and object in plain text. This notation is called *Notation 3* and is used throughout this section.

RDF data can be exposed to other applications either as downloadable data files or through a dereferencing protocol. The latter works as follows: If a semantic URI (such as `http://biographies.com/AlbertEinstein`) is accessed, the server responds with an RDF document about the entity. This way, a semantic application can gather knowledge about a given entity. Another way to access an RDF knowledge base is through *SPARQL*. SPARQL is a query language that is very similar to SQL. Thus, to ask, for example, where Einstein was born, we can send the following query to a KB

```
PREFIX b:  <http://biographies.com>
SELECT ?x WHERE { b:AlbertEinstein b:bornIn ?x.}
```

An answer to this query would bind the variable `?x` to a URI. Many KBs offer public SPARQL endpoints.

Ontology Matching. Two KBs can use different URIs to refer to the same entity. This becomes a problem if knowledge is to be queried or joined across KBs. The *Linking*

[1]Some numbers from `http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData`

Open Data Project [12] aims to establish links between URIs that represent the same entity. As we have discussed in [91], this problem has its roots in the problem of identifying duplicate entities, which is also known as record linkage, duplicate detection, or co-reference resolution. This problem has been extensively studied in both database and natural language processing areas [13; 35]. These approaches are less applicable in the context of ontologies for two reasons. First, they do not consider the formal semantics that ontologies have (such as the *subClassOf* taxonomy). Second, they focus on the alignment of entities and do not deal with the alignment of relations and classes.

There are a number of surveys and analyses that shed light on the problem of record linking in ontologies. Halpin et al. [45] provide a good overview of the problem in general. They also study difficulties of existing *sameAs*-links. These links are further analyzed by Ding et al. [34]. Glaser, Jaffri, and Millard [40] propose a framework for the management of co-reference in the Semantic Web. Hu et al. [52] provide a study on how matches look in general.

Some approaches to ontology matching have focused mainly on aligning the classes, and not the entities. Such approaches use techniques such as sense clustering [42], lexical and structural characteristics [57], or composite approaches [7]. These approaches can only align classes and do not consider the alignment of relations and entities. Other approaches in this field are [56] and [106], which derive class similarity from the similarities of the entities. Both approaches consider only the equivalence of classes and do not compute subclasses. They do not align relations or entities.

There are numerous approaches to match entities of one ontology to entities of another ontology. Ferrara, Lorusso, and Montanelli [37] introduce this problem from a philosophical point of view. Different techniques are being used, such as exploiting the terminological structure [81], logical deduction [85], declarative languages [5], relational clustering [11], or a combination of logical and numerical methods [86]. The *Sig.ma* engine [97] uses heuristics to match entities. The work in [49] introduces the concept of functionality for this purpose. The *LINDA* system [8] matches entities on large-scale datasets in a distributed manner.

The *silk* framework [105] allows specifying manual mapping rules. The *ObjectCoref* approach by Hu, Chen, and Qu [51] allows learning a mapping between the entities from training data. Hogan [48] matches entities and proposes to use these entities to compute the similarity between classes.

Some approaches address the cause of aligning both classes and entities simultaneously: the *RiMOM* [66] and *iliads* [98] systems. Both of these have been tested so far on small ontologies. The *RiMOM* system can align classes, but it cannot find *subClassOf* relationships. The *PARIS* system [91] goes one step further: It can align entities, relations, and classes at the same time. PARIS can align DBpedia and YAGO, two of the largest available KBs, in a few hours. Ultimately, the goal of ontology matching is to link the whole Semantic Web Universe into one big RDF graph.

Rule Mining. Patterns in the KB can be formalized by *rules*. For example, the following rule says that the husband of a mother is often the father of her child:

$$motherOf(m,c) \wedge marriedTo(m,f) \Rightarrow fatherOf(f,c)$$

Such rules can be mined from a KB by Inductive Logic Programming (ILP). ILP bears resemblance to association rule mining [3], but mines logical rules instead of transactions. ILP usually requires counter-examples, i.e., statements that are known to be false. RDF KBs, in contrast, contain only positive information, and no negative statements. Therefore, the ILP approaches have to simulate counter-examples when applied on Semantic Web data.

As we have summarized in [38], there are several such ILP systems. *Sherlock* [88] is an unsupervised ILP method to learn first-order Horn clauses from a set of extracted facts for a given target relation. It uses probabilistic graphical models to infer new facts and a scoring function to judge the quality of rules even in the absence of counter-examples. The WARMR system [32; 31] mines patterns that correspond to conjunctive queries. It uses a declarative language bias to reduce the search space. WARMR assumes that all information that is not in the KB is false. An extension of the system, WARMER [41], modified the approach to support a broader range of conjunctive queries and increase the efficiency of search space exploration.

The *ALEPH* system[2] is a general purpose ILP system, which implements Muggleton's Inverse Entailment algorithm [71] in Prolog. In order to learn in the absence of negative examples, *ALEPH* can run with a scoring function that uses random facts as counter-examples.

The *AMIE* system [38] was specifically designed to work on RDF KBs. It relies on the partial completeness assumption to generate counter-examples. AMIE is designed to work on large scale KBs and can mine rules from YAGO or DBpedia in a few minutes.

Another approach [80; 79] was proposed to discover causal relations in RDF-based medical data. A domain expert first defines targets and contexts of the mining process, so that the correct transactions are generated. Other approaches use rule mining to generate the schema or taxonomy of a KB. [24] applies clustering techniques based on context vectors and formal concept analysis to construct taxonomies.

Other approaches use clustering [69] and ILP-based approaches [28] to derive a schema of the KB. [43] applies clustering to identify classes of people on the FOAF network, and ILP to learn descriptions of these groups. Another example of an ILP-based approach is the DL-Learner [65], which has successfully been applied [47] to generate OWL class expressions from YAGO. As an alternative to ILP techniques, [104] propose a statistical method that does not require negative examples.

[59] proposes an algorithm for frequent pattern mining in knowledge bases represented in the formalism of DL-safe rules. Other approaches use rule mining for ontology merging and alignment [70; 29; 82]. The AROMA system [29], for instance, uses association rules on extracted terms to find subsumption relations between classes and properties of different ontologies. Another application is to generate ontologies directly from natural language text. Several approaches have been proposed that apply association rule mining [68; 103] to generate non-taxonomic relations from text. [58] considers the problem of redundant or over-generalized patterns.

In [93] typical temporal orderings between relations (e.g. $actedIn(person, film)$ happens before $wonPrize(film, award)$)

[2]`http://www.cs.ox.ac.uk/activities/machlearn/Aleph`

are induced based on the narrative order of inferred mentions of these relations, averaged across many documents. Mining is applied in this context to find verbs that express certain relations. Since RDF data can also be considered a graph with resources as vertices connected by edges labeled with predicates, mining frequent subtrees [21; 60] is another related field of research.

Scalability. The Semantic Web contains billions of statements about hundreds of different topics. All data is publicly available on the Web for free. This makes the Semantic Web one of the largest free providers of non-synthetic "Big Data". It also poses challenges to the scalability of data mining approaches. Several systems have been shown to work even on millions of statements [91; 38; 8]. Nevertheless, scalability remains an important challenge. This is because the Semantic Web is distributed, which allows it to grow fast. Most mining approaches, in contrast, are centralized.

4. KNOWLEDGE DISCOVERY FROM XML DOCUMENTS

With the massive growth of the Internet and Intranet, XML (eXtensible Markup Language) has become a ubiquitous standard for information representation and exchange [1]. Due to the simplicity and flexibility of XML, a diverse variety of applications ranging from scientific literature and technical documents to handling news summaries utilize XML in data representation. The English version of web-based free-content encyclopedia known as Wikipedia is represented as more than 3.4 million XML documents. More than 50 domain-specific languages have been developed based on XML [102], such as MovieXML for encoding movie scripts, GraphML for exchanging graph structured data, Twitter Markup Language (TML) for structuring the twitter streams and many others.

With the growing number of XML documents on the Web, it becomes essential to effectively organize these XML documents for retrieving useful information from them. The absence of an effective organization of the XML documents causes a search of the entire collection of XML documents. This search may not only be computationally expensive but my result in a poor response time for even simple queries. Knowledge discovery from XML documents, i.e., XML Mining, has been perceived as potentially one of the more effective solutions to improve XML data management for facilitating better information retrieval, data indexing, data integration and query processing [77; 74]. However, many issues arise in discovering knowledge from these types of semi-structured documents due to their heterogeneity and structural irregularity.

We present the challenges and benefits of applying mining techniques on XML documents as well as some of the popular XML data modeling and mining techniques.

4.1 XML Mining: Benefits and Challenges

Due to the inherent flexibility of XML, in both structure and semantics, modeling and mining useful information from XML documents is faced with new challenges as well as benefits. Mining of structure along with content provides new insights into the process of knowledge discovery. XML data mining methods can improve the accuracy and speed of the XML-based search engines in retrieving the relevant portions of data (1) by suggesting similar XML documents by structure and content after utilizing *XML clustering* results, and (2) by discovering the links between XML tags that occur together within XML documents after utilizing *XML frequent pattern mining* results. For example, a data mining rule can discover that the ⟨*telephone*⟩ *tags must appear within* ⟨*customer*⟩ *tags* from a collection of XML documents. This information can be used by searching only *customer* tags when executing a query about finding *telephone* details thus making the information retrieval efficient.

XML mining is an intricate process requiring both the structure features (that is, tags and their nesting therein) and the content features to consider within XML documents to derive meaningful information from them. However, many existing knowledge discovery methods of XML documents use either the structural features or the content features for finding useful information due to scalability issues [61; 73; 76; 2; 62; 33]. The sheer size and complexity of using values of both features and their relationships, in a matrix/tree/graph representation, and then processing those values to identify classes, clusters, patterns and rules turn the XML knowledge discovery process difficult and complex.

Another difficulty inherent in knowledge discovery from XML documents is flexibility in the design of XML documents. XML document authors are allowed to use user-defined tags without much structural constraint imposed with them. This results in multiple document representations for a similar content features, and different content features presented in the similar structural representations. A knowledge discovery approach should consider the heterogeneity of content and structure features within XML documents.

In summary, the strength of XML such as the capability of presenting content in a hierarchical and nested structure, flexibility in its design, capability to represent data in various forms, online availability and being distributed and autonomous, make the knowledge discovery from XML documents complex and challenging. Consider the following example. Figure 3 shows the fragments of five XML documents from the publishing domain. The XML fragments shown in d_1, d_2 and d_3 share a similar structure and so do the fragments in d_4 and d_5. If the structural similarity is considered as a criterion for clustering, two clusters are formed namely *Books* and *Conference Articles* grouping these two sets of fragments. However, it can be noted that this clustering solution has failed to distinguish between the documents (or books) of two genre and has put them all together in one cluster. For example, (d_1) and (d_2, d_3) of the *Books* cluster differ in their content. On the other hand, clustering of documents based on the content features similarity will result in two clusters, grouping d_2, d_3, d_4 and d_5 in one cluster and placing d_1 in another cluster. It can be noted that this clustering solution has failed to distinguish between *conference articles* and *books* that follow two different structures. In order to derive meaningful grouping, these fragments should be analyzed in terms of both their structural and content features similarity. For example, clustering XML documents by considering both the structural and content features together will result in three clusters, namely *Books on Data Mining*, *Books on Biology* and *Conference articles on Data Mining*. This grouping of documents can be used for effective storage and retrieval of XML documents on the Web.

d_1: $\langle Book \rangle$
$\langle Title \rangle$ On the Origin of Species $\langle /Title \rangle$
$\langle Author \rangle$ $\langle Name \rangle$ Charles Darwin $\langle /Name \rangle$ $\langle /Author \rangle$
$\langle Publisher \rangle$ $\langle Name \rangle$John Murray $\langle /Name \rangle$ $\langle /Publisher \rangle$
$\langle /Book \rangle$
d_2: $\langle Book \rangle$
$\langle Title \rangle$ Data Mining concepts and Techniques $\langle /Title \rangle$
$\langle Author \rangle$ Jiawei Han $\langle /Author \rangle$
$\langle Author \rangle$ Micheline Kamber $\langle /Author \rangle$
$\langle Publisher \rangle$ Morgan Kaufmann $\langle /Publisher \rangle$
$\langle Year \rangle$2006$\langle /Year \rangle$
$\langle /Book \rangle$
d_3: $\langle Book \rangle$
$\langle Title \rangle$ Data Mining: Practical Machine Learning Tools
and Techniques $\langle /Title \rangle$
$\langle Author \rangle$ Eibe Frank $\langle /Author \rangle$
$\langle Author \rangle$ Ian Witten $\langle /Author \rangle$
$\langle /Book \rangle$
d_4: $\langle Conference \rangle$
$\langle ConfTitle \rangle$ Survey of Data Mining Techniques $\langle /ConfTitle \rangle$
$\langle ConfAuthor \rangle$ John Smith $\langle /ConfAuthor \rangle$
$\langle ConfYear \rangle$ 2007 $\langle /ConfYear \rangle$
$\langle ConfName \rangle$ SIAM International Conference $\langle /ConfName \rangle$
$\langle /Conference \rangle$
d_5: $\langle Conference \rangle$
$\langle ConfTitle \rangle$ Combining Content and Structure for
XML document Similarities $\langle /ConfTitle \rangle$
$\langle ConfAuthor \rangle$ Richi Nayak $\langle /ConfAuthor \rangle$
$\langle ConfLoc \rangle$ Adelaide, Australia $\langle /ConfLoc \rangle$
$\langle ConfName \rangle$ Australian Data Mining Conference $\langle /ConfName \rangle$
$\langle /Conference \rangle$

Figure 3: A Sample XML data set containing 5 XML document fragments.

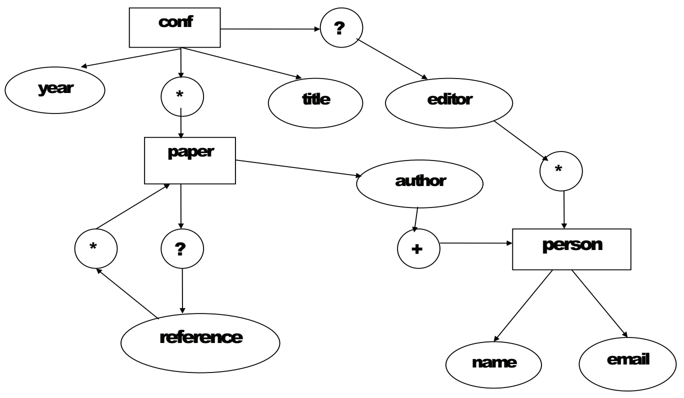

Figure 4: Graph representation of a schema

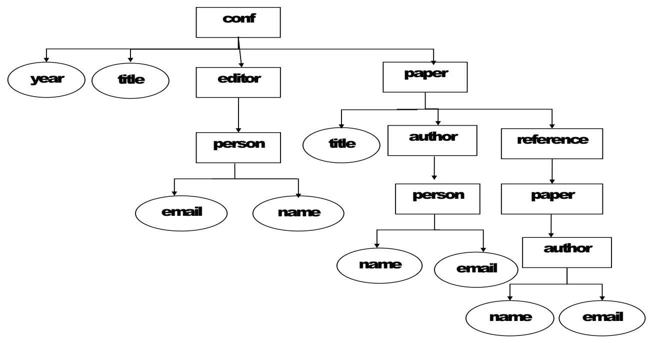

Figure 5: Tree representation of an XML document

4.2 XML Data Modeling for Knowledge Discovery

Before the useful information from XML documents can be mined, it is necessary that they are represented in a format appropriate to a mining algorithm. The common data representations are matrix, tree and graph models. The graph representation is commonly used in mining the XML schema definitions documents. If a schema is not available with documents, the structural features can be extracted from them to be represented as graphs or trees. Usually it is a straightforward process of parsing the document to map its structure to a data model. There also exist some tools that can infer a schema given a number of documents conforming to it [39; 74]. Figure 4 shows the graph representation of a schema document. It can be noted that there is a cyclic reference to the tag *paper* from the tag *reference*.

Tree representation, on the other hand, is commonly used for XML documents. A tree is an acyclic graph with a root node of zero indegree. The leaf nodes of the tree contain the content enclosed within that node. An example of the rooted, ordered, labeled tree representation corresponding to an XML document is shown in Figure 5.

The matrix representation of XML documents uses the vector-space model [87] to present the content and structure features of the documents. The structure features of an XML document is represented as a collection of distinct paths in the set of documents. The structure features of all the documents in the dataset can be put together as a path-document matrix, $PD_{m \times n}$. Each cell in matrix $PD_{m \times n}$, can be the (weighted) frequency of a distinct path appearing in a document using the weighting schemes such as binary, term frequency, $tf \times idf$ or BM25 [87]. Similarly a term-document matrix, $TD_{l \times n}$ can be constructed after processing the content features of documents. The structural features of the dataset in Figure 3, containing 5 XML fragments (d_1, d_2, d_3, d_4, d_5) can be shown as the path-document matrix in Figure 6. The matrix representation of a XML dataset risks losing some of the hierarchical features of the dataset, however, it benefits by allowing the use of advanced matrix factorization methods such as latent semantic modeling [63] and non-Negative matrix factorization methods [64].

4.3 XML Mining Techniques

Once the XML documents are modeled, appropriate data mining techniques can be applied to data models for knowledge discovery. Some examples of knowledge discovery from XML documents are mining frequent sub-tree/sub-graph patterns, grouping and classifying documents/schemas, mining XML queries for efficient processing and schema discovery from a number of XML documents. We discuss XML data clustering and XML frequent tree mining techniques here.

path/doc	d_1	d_2	d_3	d_4	d_5
Book/Title	1	1	1		
Book/Author	1	2	2		
Book/Publisher	1	1			
Book/Year		1			
Conference/ConfTitle				1	1
Conference/ConfAuthor				1	1
Conference/ConfName				1	1
Conference/ConfYear				1	1
Conference/ConfLoc				1	

Figure 6: A matrix PD representing the structure features of the dataset

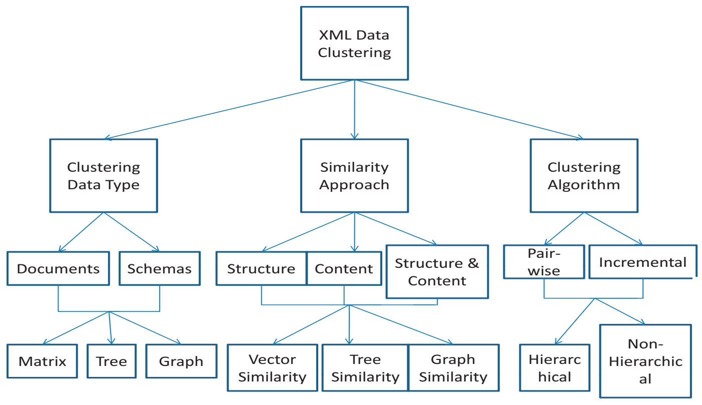

Figure 7: An XML Clustering Taxonomy

XML Data Clustering. Given a set of XML documents D, a clustering solution $C = (C_1, C_2, ...C_q)$ is a disjoint partition of $(D_1, D_2, ...D_n)$. C_i denotes a cluster in the clustering solution and D_i denotes an XML document represented as a tree or graph or vector-space model. There are three main elements of a XML clustering method as shown in Figure 7. The first element is the type of XML data: documents or/and schemas.

The second element of a clustering method is the similarity approach that determines how similar two documents are. The similarity approach predominantly depends on (1) the data models that are used in representation of the XML data and (2) the type of features these data models are representing: structural or content or both. If the content and/or structural features of the documents are represented in a vector-space model, metric distances induced by norms are typically used. The best-known examples are the L1 (Manhattan) and L2 (Euclidean) distances. Other popular similarity measures used are the cosine ($cos(v1, v2) = v1v2/|v1||v2|$), Dice ($Dice(v1, v2) = 2v1v2/|v1|^2|v2|^2$), and Jaccard ($Jac(v1, v2) = v1v2/|v1|^2|v2|^2 - v1v2$) coefficients. Recently the semantic models such as Latent Semantic Kernels have also been used in calculating the content similarity between XML documents [95]. The similarity coefficients based on structural features and content features are calculated independently, and then combined weighted linearly or non-linearly to form a single measure. If a document structure is represented using tree or graph models, the tree-edit distance methods or specific path-similarity coefficients considering the contextual positions of tags are employed to determine similarity between two graphs or trees [75; 76; 73].

The third element of a clustering method is the clustering algorithm employed to take the similarity coefficients into consideration to group the documents. There are two basic types of algorithms used. The first and most commonly used type of algorithms are based on pair-wise similarity that is determined between each pair of documents/schemas in the dataset. A pair-wise similarity matrix is generated for the dataset, on which a centroid approach such as the K-means algorithm or a hierarchical approach such as agglomerative and divisive algorithms can be applied to find groupings within documents/schemas. On the other hand, an incremental type of clustering algorithm does not need to compute a pair-wise similarity matrix. An incremental clustering algorithm defines the clustering criterion function on the cluster level to optimize the cluster parameters. Each XML data is compared against the existing clusters instead of comparing to another individual XML data. Since the computation of the global metrics is much faster than that of the pair-wise similarities, an incremental clustering algorithm is efficient and scalable to a very large dataset [73].

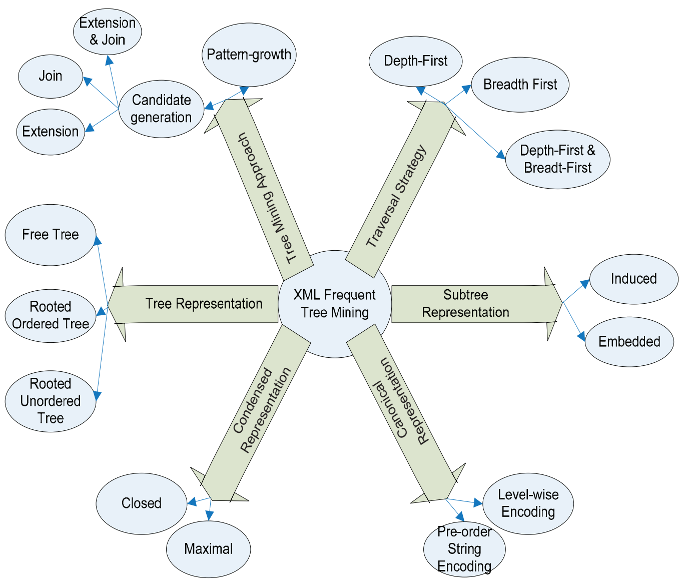

Figure 8: An XML Frequent Tree Mining Taxonomy

XML Frequent Pattern Mining. A major difference between frequent itemset mining and XML frequent pattern mining is consideration of ordering among the items in the latter. The ordering of items in XML documents depends on the level of nesting among the tags. Given a set of XML documents, the frequent pattern set contains patterns that have support or frequency greater than $minThreshold$, a user-defined minimum support that indicates the least number of documents in which the pattern appears in. The modeling of XML documents forms the perception of patterns. For instance, if the XML document set is modeled as matrix (that is each row represents a document and each column represents the tag in the XML document) then the frequent patterns are a list of frequently occurring tags. The matrix representation of documents allows the standard itemset mining algorithms such as Apriori [3] to extract frequent patterns from the XML dataset; however, it ignores the hierarchical relationships between tags.

The matrix models of data representation for frequent pattern mining may result in providing inaccurate results. Consider, for example, the following two fragments: ($\langle vessel \rangle \langle craft \rangle$ boat $\langle /craft \rangle \langle /vessel \rangle$) in XML document A and the fragment ($\langle occupation \rangle \langle craft \rangle$ boat *building* $\langle /craft \rangle \langle /occupation \rangle$) in XML document B. The tag *"craft"* can be considered frequent if *minThreshold* is 2 or more, and the hierarchical relationships are not considered as in this case. However the tag $\langle craft \rangle$ is not frequent as its parents are different, the former refers to a vessel and the latter to an occupation. Hence, to avoid inaccurate results, it is essential to consider the hierarchical relationships among the tags and not just the tag name.

Use of XML frequent pattern mining provides the probability of a tag having a particular meaning. For example, a mining rule inferred from a collection of XML documents is *"80% of the time, if an XML document contains a $\langle craft \rangle$ tag then it also contains a $\langle driver \rangle$ tag"*. Such a rule helps determine the appropriate interpretation for homographic tags such as $\langle craft \rangle$ and can improve information retrieval by distinguishing two documents who appears similar due to the presence of homographic tags.

If a XML document is modeled as a tree or a graph preserving the hierarchical information then the frequent patterns are frequent subtrees or frequent subgraphs. A sub-tree or sub-graph preserves the predecessor relationship among the nodes of the document tree/graph. In spite of the advancements in graph-mining [53; 54], these algorithms have often been criticized for producing more generic results and incurring excessive time to generate results [116]. Consequently researchers have shifted their attention to XML frequent tree mining algorithms by using trees instead of graphs for representing XML data [21]. Figure 8 represents various factors important for XML frequent tree mining algorithms, see [62] for more information.

4.4 Scalability

With the growing importance of XML in document representation, efficient and effective knowledge discovery methods need to be devised. Scalability to the very large datasets should become a basic functionality of the newly developed methods. Advances in low cost high-powered servers now just begin to facilitate indexing and mining, such as clustering and links, at the sub-document level. This means that the number of objects considered can be several orders of magnitude larger than the number of documents in a collection. State-of-the-art research into Web scale collections is dealing with the order of one billion documents (e.g., ClueWeb 2012 [20]), but it is now possible to process sub-document parts, thereby effectively dealing with one to three order of magnitude larger collections (in the number of objects indexed/clustered/linked). Is the matrix/tree/graph representation sufficient for big data? In the big data context, another representation or an extension to the existing ones, such as the sequence representation or XML structural summaries, is necessary to face the performance requirements. There may not be the need of representing each keyword in these representations. Latent Dirichlet Allocation and Non-negative matrix factorization can be used to generate concepts in the dataset. So the document contents can be represented as topics in order to deal with scalability. Data mining techniques greatly help in representing the big data, by showing the patterns rather than raw data. For example, clusters can be used in representing meaningful summaries of the data. A cluster can be represented in many ways according to the computation capabilities such as the largest tree in the cluster, or the frequent trees among all trees in the cluster, or a cluster level structure containing all elements of each tree at each level. Similarly, frequent trees can be used to represent the common structures that are inherent in the raw data.

Hence, given this description, we can see that XML mining techniques facilitate the discovery of knowledge from the Web data consisting of plain text, numbers and complex data types stored in a seamless manner. These techniques focus on both the structure and the content of the data, being potentially interesting to users trying to discover knowledge from different aspects. For example, considering the works of Einstein, XML structure mining would enable us to identify the common tags that are used to express his work and how are they related. These aspects can be useful to group relevant papers on specific topics and search them. XML content mining would enable the identification of the common themes that arise from the works of Einstein. These aspects can be useful to discover similarities and differences between his research and that of other scientists working on related topics.

With this detailed background of XML modeling and mining, we next proceed to explain DSMLs which grow on the basis of XML.

5. DSML DATA STORAGE AND MINING

A domain-specific markup language, abbreviated as DSML, is a language that follows the XML syntax and encompasses the semantics of the given domain. Such languages offer several advantages. Markup languages designed using XML tag sets to capture the needs of specific domains make it easier for users of the respective domains to exchange information across a common platform. Moreover, this facilitates storage and publishing of information worldwide without having to use a medium of conversion. It also enhances the retrieval and analysis of data from the users' angle, e.g., embedded formulae, as will be illustrated with examples here. Furthermore, this facilitates the use of data mining techniques such as association rules and clustering to discover knowledge from the XML-based structured data in a given domain from an application standpoint as elaborated later.

Accordingly, many domain-specific markup languages are designed using special XML tag sets. Standardization bodies and research communities may extend these to include additional semantics of areas within and related to the domain. Such markup languages thus serve as communication media for the targeted users such as industries, universities, publishers, research laboratories and others.

5.1 DSML Examples

We present a few examples of domain-specific markup languages existing in the literature. We will later provide more specific examples pertaining to the use of knowledge discovery techniques in conjunction with some of these DSMLs.

MML. This is the Medical Markup Language [44] developed in Japan to create a set of standards by which medical data can be stored, accessed and exchanged. The following MML module contents are defined: patient information, health insurance information, diagnosis information, lifestyle information, basic clinic information, particular information at the time of first visit, progress course information, surgery record information and clinical summary information [44]. They are of use to primary care physicians, general surgeons, their patients and related entities. However, specific information, for example opthalmological details such as eye-diseases and spectacle prescriptions cannot be stored using these general tags. Thus, a more specific markup language can potentially be developed as needed within the context of MML to include the semantics of opthalmology.

MatML. This is the acronym for Materials Markup Language [10] developed by the National Institute of Standards and Technology, NIST in the USA. MatML serves as the XML for materials property data [10; 99]. MatML elements are bulk details, component details, metadata, graphs and glossary of terms. They have sub-elements and attributes for storage of information related to properties of materials such as metals, ceramics and plastics. For example, the chemical composition of an alloy would be stored under component details. Specific markups can be defined if required as extensions to MatML to capture sub-areas. For example, one sub-area is the Heat Treating of Materials which involves the controlled heating and cooling of materials to achieve desired mechanical and thermal properties [96]. Quenching is the rapid cooling of the material in a liquid or gas medium [96] and forms an important step of the Heat Treating processes. In order to store data on these processes, a markup language called QuenchML [100] has been developed by the Center for Heat Treating Excellence, an industry-university consortium.

MathML. A product of the W3C Math working group is MathML, the Mathematical Markup Language [17]. This is a specification for describing mathematics as a basis for machine to machine communication. It provides much needed foundation for the inclusion of mathematical equations in Web pages. Thus, a variety of formulae can be included within the MathML tags, and they would be retrieved during a Web search. MathML provides the tag set for including several expressions ranging from simple additions and subtractions to more complicated matrix operations. A fundamental challenge in defining a markup language for mathematics on the Web is reconciling the need to encode both the presentation of a mathematical notation and the content of the mathematical idea or object which it represents. This is done by using a *presentation markup* tag set pertaining to display of the notation, and a *content markup* tag set pertaining to executing the actual mathematical idea.

Other Languages. More examples of XML-based domain-specific markup include the following. CML, the Chemical Markup Language [72] serves as a powerful medium to manage molecular and technical information in chemistry. CML is object-oriented, being based on Java and XML, and spans a fairly wide range of chemical disciplines. ModelML, the Model Markup Language [117] provides a descriptive framework of object-oriented models to be synthesized. It thereby enhances the development of automatic model synthesis by decoupling the knowledge per se from the tools to be built. SSML, the Speech Synthesis Markup Language [94] stores data related to speech and pronunciation. Its input is text-based with no constraints on word usage and it has additional information included pertaining to speech, e.g., annotations with markers to specify features such as emphasis, particular speech styles, or the beginning of new topics. With this introduction, we now proceed to discuss how DSMLs are useful.

5.2 Usefulness of DSMLs

Domain-specific markup languages are very useful in retrieving information in a meaningful fashion due to their semantic tag sets, while also providing worldwide access in a convenient manner due to their XML-based syntax.

It is very important to note the specific role played by DSMLs here. In the absence of DSMLs, information such as a formula may have been be treated as an embedded picture while storing and retrieving Web information. However, if this information is stored using MathML, it would actually be preserved as a formula and get retrieved accordingly, due to the availability of the presentation and content markups in MathML.

We illustrate this with respect to the famous equation on Einstein's Theory of Relativity, $E = mc^2$, where E is energy, m is mass and c is speed of light in a vacuum. Using MathML's presentation markup, this is stored as follows.

```
⟨mrow⟩
    ⟨mi⟩E⟨/mi⟩
    ⟨mo⟩=⟨/mo⟩
    ⟨mi⟩m⟨/mi⟩
    ⟨msup⟩
            ⟨mi⟩c⟨/mi⟩
            ⟨mn⟩2⟨/mn⟩
    ⟨/msup⟩
⟨/mrow⟩
```

This example demonstrates some presentation elements. The first element is *mrow* used to denote a row of horizontally aligned material. The *mi* element is used for displaying variables, such as E and m, while the *mo* element is used for displaying operators, in this case, the = operator. The *msup* element is for superscripts and takes two arguments, the base expression, which is c here (enclosed in *mi* tags since it is a variable), and the exponent expression which is 2 here, and is enclosed in *mn* tags, because *mn* denotes a constant in MathML presentation syntax.

Using content markup, the equation $E = mc^2$ is stored as shown in the code snippet below.

```
⟨mrow⟩
⟨apply⟩
    ⟨eq/⟩
    ⟨ci⟩E⟨/ci⟩
    ⟨apply⟩
```

```
        ⟨times/⟩
        ⟨ci⟩m⟨/ci⟩
        ⟨apply⟩
            ⟨power/⟩
            ⟨ci⟩c⟨/ci⟩
            ⟨cn⟩2⟨/cn⟩
        ⟨/apply⟩
    ⟨/apply⟩
⟨/apply⟩
⟨/mrow⟩
```

Here, the *apply* content element means that an operator is applied to an expression. In this example, the operators applied are *equals* denoted by the *eq/* content element, *multiplication* denoted by the *times/* content element and *exponentiation* denoted by the *power/* content element. Note that the operators take arguments, the order being particularly significant in the case of the power operator. However, the order of the children is crucial in the use of the *apply* content element, since its first child, the operator, takes as an argument list the remaining operands and other nested operators. Observe also, the use of the *ci* element to mark up the variables E, m and c and the use of the *cn* element to mark up the constant 2.

Depending on the nature of the application, either one of or both of these markups can be useful, in order to help discover the appropriate information from Web documents. For example, legacy data on the works of Albert Einstein would probably be best translated into pure presentation markup. On the other hand, some mathematical applications and pedagogical tools would be more likely to prefer the content-based markup. Also, certain applications such as game development and decision support would be somewhere in between these extremes, where it would be advisable to use a mixture of both presentation and content markup.

In all these applications, MathML data, like that with any DSML, can be embedded in a regular XML document that can contain information on the explanation of the theory, along with the formulae. Thus, semantically the storage and retrieval capabilities of MathML are far better than that of HTML in the context of mathematics. Such storage also sets the stage for applying knowledge discovery techniques in a more meaningful manner analogous to the way humans analyze data.

Consider another example in SSML (Speech Synthesis Markup Language). The annotated information in the tags related to pronunciation such as emphasis, styles etc. is extremely useful in constructing speech from plain text. In the absence of this annotation, these important vocal aspects pertaining to speech would be ignored and written text would be treated no differently from oral speech when being conveyed as a document. Thus, semi-structured storage with the presence of XML-based tags conveying additional knowledge in SSML proves useful.

Similar arguments on the importance of XML-based semi-structured storage preserving domain semantics can be applied to other DSMLs such as MML and CML. With this discussion, we now delve deeper into mining XML data in conjunction with DSMLs.

5.3 Association Rule Mining with DSMLs

The mining of association rules for XML documents extends the concept of frequent pattern mining in XML. As widely known to the data mining community, association rule mining is the discovery of associations of the type $A \Rightarrow B$, where A is called the antecedent and B is called the consequent, both A and B being features of the dataset. A popular algorithm for association rule mining is the Apriori algorithm [3] which uses prior information about the frequency of occurrence of items in the given data set to discover knowledge about their likelihood of occurring together. Interestingness measures for association rules are Rule Confidence and Rule Support calculated as follows.

$Rule\ Confidence = P(B/A) = count(A \wedge B)/count(A)$

$Rule\ Support = P(A \wedge B) = count(A \wedge B)/count(set)$

Using these interestingness measures of rule confidence and rule support along with and suitable thresholds for minimum confidence and minimum support, the Apriori algorithm mines association rules [3].

We now consider the mining of association rules in the context of domain-specific markup languages. If data in the given domain is stored using the respective DSML, it is convenient for deriving the corresponding associations. Let us apply the Apriori algorithm for the discovery of association rules from medical data.

Consider the following example with a random sample of data stored using a semi-structured format with tags as in the medical markup language MML.

⟨*Smoking*⟩ yes ⟨/*Smoking*⟩ in 44/52 instances
⟨*LungCancer*⟩ yes ⟨/*LungCancer*⟩ in 41/52 instances
  38 of these in common with ⟨*Smoking*⟩
Association Rule: $Smoking = yes \Rightarrow LungCancer = yes$
  $Rule\ Confidence = 38/44 = 86.36\%$
  $Rule\ Support = 38/52 = 73.07\%$

In this example, we refer to publications on the subject of lung cancer with documentation on specific cases. Here, 44 cases of patients are found to be smoking among a total of 52 cases considered. Furthermore, 41 out of all these 52 cases refer to patients diagnosed with lung cancer, 38 of which are common with smoking patients. Applying the Apriori algorithm over this data stored using semantic tags, we get a rule $Smoking = yes \Rightarrow LungCancer = yes$ with a rule confidence of $38/44 = 86.36\%$ and rule support of $38/52 = 73.07\%$ using the given formulae. The data for such analysis would often be found in text sources with documentation on medical cases. The use of MML to store this textual data in semi-structured formats facilitates rule derivation.

We emphasize the advantage of the method described above for rule discovery over text sources. The plain text in such formats can be converted to XML-based formats, i.e., semi-structured text using the markup language with the help of natural language processing tools. This semi-structured text can then be further preprocessed to a form suitable for rule mining. An algorithm such as Apriori [3] can then be used to extract rules from these text sources after conversion.

Note that the use of semantic tags in DSMLs such as MML or MatML in the corresponding domains facilitates the capturing of relevant information from the plain text sources. The DSML is crucial in guiding the conversion of plain text into XML based semi-structured text by extracting suitable information corresponding to the given tags that is of interest to domain-specific users. On the contrary, in the absence of a DSML, applying association rules over plain text sources by simple removal of stem words could lead to obvious rules such as *doctor* ⇒ *patient* in the medical domain, and *experiment* ⇒ *result* in the Materials Science domain. This is due to the frequent occurrences of these terms within related sources of literature. There would also be extreme lack of efficiency in mining over a seemingly infinite number of terms. The use of DSMLs serves to filter out unwanted terms and capture only the relevant ones for the discovery of meaningful association rules, and also makes the mining more efficient.

5.4 Clustering with DSMLs

Performing XML clustering in conjunction with domain-specific markup languages assists in organizing information stored in the documents more systematically. Suppose there are a number of documents stored in an organization using DSMLs, these documents can be grouped according to their thematic features or their structural features. For example, given a number of instances of Materials Science experimental data, XML clustering can group all instances pertaining to agitation of a cooling medium, distortion of a part and so on in a rapid cooling process, using the corresponding semantic tags such as ⟨*agitation*⟩ and ⟨*distortion*⟩ respectively, provided by a given DSML such as the Materials Markup Language MatML or its extension QuenchML that captures the semantics of heat treating and rapid cooling processes.

Likewise, with medical data, XML clustering using markups can be used to group patient documents together in categories based on the occurrence of diseases as identified by the semantic tags in the medical markup language MML. Similarly, considering the example on Einstein's works introduced earlier, performing XML clustering on the data stored using the presentation tag set of MathML can be useful in grouping all documents relevant to specific equations, such as the one on relativity. Note again that DSML data can be embedded within an XML document, thus the XML tags themselves would serve to provide a categorization based on other features such as titles and authors. A finer level of granularity would be provided by the respective DSML tags. Thus in general, the semantic tags in the domain-specific markup language can guide the clustering process. These smaller groups of information can be easily visualized for further analysis.

The benefit of using XML clustering along with such markup languages for the information stored in the documents is that it can leverage on both the *structure* and *content* information inherent in those documents. Sometimes users present similar information using different tags and structure. By using XML clustering with domain-specific markup languages, this information would still be grouped together even if it looks and feels different. Likewise, if different content is represented in same hierarchical structure, XML clustering performed with domain-specific markup languages could differentiate between those datasets.

The knowledge discovered by applying such techniques over XML documents in conjunction with DSMLs can be useful in several applications, e.g., expert systems, simulation tools and intelligent tutors in the corresponding domains. For example, expert systems for medical diagnostics can be designed by mining data from relevant sources of literature in medicine guided by MML. Simulation tools in domains such as Materials Science can be enhanced by the discovery of association rules from MatML based semi-structured data. Intelligent tutors in Mathematics can be developed and enhanced using the knowledge discovered from MathML sources. Likewise, various pedagogical tools and other software applications would benefit from the knowledge acquired by mining over various sources of literature in the concerned scientific domains.

5.5 Perspective on Scalability

The issue of scalability in DSMLs is often debatable. Some language developers argue that it is good to construct the DSML tag set such that it is relatively easy for users of related domains to augment it with additional semantics. For example, QuenchML (Quenching Markup Language) [100] has been proposed as an extension to MatML (Materials Markup Language) [10] to capture the semantics of one of its sub-areas called Quenching (a crucial heat treating process). MatML per se is the XML for materials property data. It has primarily been designed to store various material properties but does not include the details of all the materials processes. Thus, semantic extensions such as QuenchML can be developed as needed. The MatML tag set facilitates the inclusion of additional semantics, providing several common elements that can be shared and reused.

On the other hand, there are counter-arguments that too much scalability presents an obstacle in the path of standardization. Standards bodies are often not in favor of an existing DSML standard being updated frequently. Moreover, standardization can take a long time subject to approval by such standards bodies, e.g., NIST (National Institute of Standards and Technology, USA). However, some developers argue further in response to this problem. They state that even if a DSML extension is not standardized, its tag set can still be made available to users for information exchange. This serves as a lingua franca in the respective sub-area of the domain, which is desirable.

Another aspect of scalability is with respect to big data. Here, we claim that DSMLs would adapt well to data with higher orders of magnitude provided the required physical storage is available. This applies to storing as well as mining big data. More efficient algorithms would be needed to handle DSML mining with big data. However, the tag set may not necessarily require extension to cater to issues of scale. In the event that it does need extension, the arguments portrayed above on the pros and cons of scaling would be valid for big data as well. Furthermore, the issues that apply to XML scalability are also to be taken into account with DSMLs, since they grow on the basis of XML. On the whole, after considering various aspects, we state that scalability is a desired property of DSMLs and that they could potentially scale well to handle big data.

6. OUTLOOK AND CHALLENGES

We now present a general outlook on the Web developments addressed in our survey article. We also list some challenges and open research questions.

Let us first look at the problem of knowledge discovery on the deep Web. One of the main challenges is the question of how to get the precise semantics of a service. Consider a service that, given a person as an input, returns a year as an output. How can we know whether this year is the birth date, the death date, or the graduation date? This is largely an open problem, though some partial solutions are discussed in [89]. Ideally, we would want to obtain a fully semantic description of a deep Web service, using for instance the semantic language for Web services, OWL-S. Second, dealing with complex forms, especially when there are dependencies between form fields, is a fully open problem in its generality. An interesting source of information could be the Javascript code that validates the data entered into a form. This code could help derive knowledge on the form itself. Static analysis techniques could be applied to the Javascript code to determine which fields of a form are required, for instance. Note, finally, that much of what we discussed is actually applicable not only to deep Web applications, but also to Web 2.0/AJAX applications. An autocomplete feature, for example, can be seen as functionally equivalent to a Web form that takes as input a sub-word and returns a full word. More complex AJAX applications might be dealt with as deep Web sources.

In summary, although there are techniques to derive the type of input parameters and output records of a Web form, it is still challenging to get the precise semantics of a service. Moreover, dealing with complex forms (such as those used to access specialized scientific databases), especially when there are dependencies between form fields, required and optional fields, etc., is a fully open problem by itself. Finally, a knowledge discovery system for the hidden Web that would not cover specific Web sites or a specific domain of interest but would work at the scale of the whole Web is still to be constructed.

The Semantic Web has attracted interest from scientists and industry alike. Numerous projects [107] transfer Semantic Web technologies to organizations and businesses. The number of semantic knowledge bases is continuously growing. New applications, such as enhancing maps with semantic data, or extracting RDF data from natural language text, are coming to life.

The Semantic Web is still relatively young. Numerous challenges still wait to be solved. One of the main open research questions is the reconciliation of different semantic conceptualizations in different ontologies. Another challenge is the expansion of the Semantic Web, be it through community work, by converting existing databases into RDF, or by Information Extraction. Reasoning on Web scale is likewise still an open issue. How can we apply automated reasoners on huge, potentially noisy data sets? While the challenges on the Semantic Web resemble challenges in other research areas, they come with additional twists due to the semantics, scale, and distribution of the data.

Let us now look at the area of XML. With the growing importance of XML in document representation, efficient and effective knowledge discovery methods need to be devised. Scalability to very large datasets is one of the main desiderata for such methods. It is important to address the issue of whether the matrix/tree/graph representation is still sufficient. In the context of large scale data, another representation or an extension to the existing ones, such as the sequence representation or XML structural summaries, could be necessary to satisfy performance requirements. Moreover, future XML mining techniques will have to consider both the structure and the content features, in order to deal with the heterogeneity of the XML documents.

Finally, let us consider DSMLs. Since a DSML follows the XML syntax and captures the semantics of the given domain, it often serves as a non-proprietary Esperanto for targeted user bodies. New DSMLs are being developed and existing ones are subjected to semantic extensions. This calls for further research considering the trade-off of extensibility versus standardization, the development of DSML-based querying and mining packages, and the design of user-friendly interfaces for DSMLs.

In the area of XML and DSMLs, considerable future work can emerge from joint research in these areas. A major challenge here is to effectively model both structure and content features for XML documents for appropriately mining the data using DSMLs. It is also challenging to combine structure and content features in different types of data models. Adequately harnessing DSMLs to integrate background knowledge of specific domains into various XML mining algorithms is yet another challenging task. Furthermore, it might become necessary to develop new standards that use the synergy between XML and DSMLs. This, in turn, would present additional research challenges with respect to standardization and extensibility.

We believe that more research in the areas of the hidden Web, the Semantic Web, XML, and domain-specific markup languages can further enhance the field of knowledge discovery on the Web.

7. CONCLUSIONS

This survey article gives an overview of harvesting various forms of Web information. It addresses developments in the areas of the deep Web, the Semantic Web, XML and DSMLs, explaining how these can be harvested for the retrieval of useful information from the Web. Given the irony that humans produce far more data than they can ever use, the development of knowledge discovery methods must keep pace with the development of Web technology itself. Research that cuts across two or more of these areas is particularly challenging and interesting. We believe that new technologies for these evolving areas of the Web could make the Web even more powerful and useful.

8. REFERENCES

[1] Abiteboul, S., Buneman, P., Suciu, D.: *Data on the Web: From Relations to Semistructured Data and XML*. California: Morgan Kaumann, 2000.

[2] Aggarwal, C. C., Ta, N., Wang, J., Feng, J., Zaki, M.J: Xproj: a framework for projected structural clustering of XML documents. In *Proc. KDD*, pp. 46–55, 2007.

[3] Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In *Proc. SIGMOD*, pp. 207–216, 1993.

[4] Antoniou, G., F. van Harmelen. *A Semantic Web Primer (Cooperative Information Systems)*. The MIT Press, April 2004.

[5] Arasu, A., Garcia-Molina, H.: Extracting structured data from Web pages. In *Proc. SIGMOD*, pp. 337–348, 2003.

[6] Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In *Proc. ISWC*, pp. 722–735, 2007.

[7] Aumueller, D., Do, H. H., Massmann, S., Rahm, E. Schema and ontology matching with COMA++. In *Proc. SIGMOD*, pages 906–908, 2005.

[8] Böhm, C., de Melo, G., Naumann, F., Weikum, G.: LINDA: Distributed web-of-data-scale entity matching. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM)*, New York, NY, USA, 2012. ACM.

[9] Barbosa, L., Freire, J.: Searching for hidden-Web databases. In *Proc. WebDB*, pp. 1–6, 2005.

[10] Begley, E.F.: *MatML Version 3.0 Schema*. NIST 6939, National Institute of Standards and Technology Report, USA, Jan 2003.

[11] Bhattacharya, I., Getoor, L. Collective entity resolution in relational data. *ACM TKDD*, 1, 03 2007.

[12] Bizer, C., Heath, T., Idehen, K., Berners-Lee, T.: Linked data on the web (LDOW2008). In *Proc. WWW*, 2008, pp. 1265–1266.

[13] Bleiholder, J., Naumann, F. Data fusion. *ACM Computing Surveys*, 41(1), 2008.

[14] Boag, S., Fernandez, M., Florescu, D., Robie J., Simeon, J.: *XQuery 1.0: An XML Query Language*. W3C Working Draft, Nov 2003.

[15] Bray, T., Hollander, D., Layman, A., Tobin, R.: *Namespaces in XML 1.0 (Second Edition)*. W3C Recommendation, Aug 2006.

[16] BrightPlanet: *The deep Web: Surfacing hidden value*. White Paper, Jul 2000.

[17] Carlisle, D., Ion, P., Miner, R., Poppelier, N.: *Mathematical Markup Language (MathML)*. World Wide Web Consortium, 2001.

[18] Caverlee, J., Liu, L., Buttler, D.: Probe, cluster, and discover: Focused extraction of qa-pagelets from the deep Web. In *Proc. ICDE*, pp. 103–115, 2004.

[19] Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[20] Charles, L., Clarke, A., Craswell, N., Soboroff, I., and Voorhees, E.: *Overview of the TREC Web Track*, 2011.

[21] Chi, Y., Nijssen, S., Muntz, R.R., Kok, J.N.: Frequent subtree mining — an overview. *Fundamenta Informaticae*, 66(1-2):161–228, 2005.

[22] Chuang, S.L., Chang, K.C., Zhai, C.: Context-aware wrapping: Synchronized data extraction. In *Proc. VLDB*, pp. 699–710, 2007.

[23] Chang, K.C., He, B., Zhang, Z.: Toward large scale integration: Building a metaquerier over databases on the Web. In *Proc. CIDR*, pp. 44–55, 2005.

[24] Cimiano, P., Hotho, A., Staab., S.: Comparing Conceptual, Divisive and Agglomerative Clustering for Learning Taxonomies from Text. In *ECAI '04*. IOS Press, 2004.

[25] Clark, J.: *XSL Transformations (XSLT)*. W3C Recommendation, Nov 1999.

[26] Clark, J., DeRose, S.: *XML Path Language (XPath)*. W3C Recommendation, Nov 1999.

[27] Crescenzi, V., Mecca, G., Merialdo, P.: Roadrunner: Towards automatic data extraction from large Web sites. In *Proc. VLDB*, pp. 109–118, 2001.

[28] d'Amato, C., Fanizzi, N., Esposito, F.: Inductive learning for the Semantic Web: What does it buy? *Semant. web*, 1(1,2), Apr. 2010.

[29] David, J., Guillet, F., Briand, H.: Association Rule Ontology Matching Approach. *Int. J. Semantic Web Inf. Syst.*, 3(2), 2007.

[30] Davidson, S., Fan, W., Hara, C., Qin, J.: Propagating XML Constraints to Relations. In *Proc. ICDE*, pp. 543 - 552, 2003.

[31] Dehaspe, L., Toironen, H.: Discovery of relational association rules. In *Relational Data Mining*. Springer-Verlag New York, Inc., 2000.

[32] Dehaspe, L., Toivonen, H.: Discovery of frequent DATALOG patterns. *Data Min. Knowl. Discov.*, 3(1), Mar. 1999.

[33] Denoyer, L., Gallinari, P..: Report on the xml mining track at inex 2007. *ACM SIGIR Forum*, pp. 22 - 28, 2008.

[34] Ding, L., Shinavier, J., Shangguan, Z., McGuinness D. L.: SameAs networks and beyond: Analyzing deployment status and implications of owl:sameAs in linked data. In *Proc. ISWC*, pages 142–147, 2010.

[35] Elmagarmid, A., Ipeirotis, P., Verykios, V. Duplicate record detection: A survey. *IEEE TKDE*, 19(1):1–16, 2007.

[36] Erik, W., Robert, J.: Xml fever. *Communications of the ACM*, 51(7):40-46, 2003.

[37] Ferrara, A., Lorusso, D., Montanelli, S.: Automatic identity recognition in the semantic web. In *Proc. IRSW*, 2008.

[38] Galarraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases. In *WWW 2013*, 2013.

[39] Garofalakis, M.N., Gionis, A., Rastogi, R., Seshadri, S., Shim, K.: Xtract: A system for extracting document type descriptors from xml documents. In *Proc. SIGMOD*, pp. 165 - 176, 2000.

[40] Glaser, H., Jaffri, A., Millard, I.: Managing co-reference on the semantic Web. In *Proc. LDOW*, 2009.

[41] Goethals, B., Van den Bussche, J.: Relational Association Rules: Getting WARMER. In *Pattern Detection and Discovery*, volume 2447. Springer Berlin / Heidelberg, 2002.

[42] Gracia, J., d'Aquin, M. Mena, E.: Large scale integration of senses for the semantic Web. In *Proc. WWW*, pages 611–620, 2009.

[43] Grimnes, G.A., Edwards, P., Preece, A.D.: Learning Meta-descriptions of the FOAF Network. In *ISWC'04*, 2004.

[44] Guo, J., Araki, K., Tanaka, K., Sato, J., Suzuki, M., Takada, A., Suzuki, T., Nakashima, Y., Yoshihara, H.: MML (Medical Markup Language) Version 2.3 - XML based Standard for Medical Data Exchange/Storage. *Journal of Medical Systems*, 27(4):357–366, 2003.

[45] Halpin, H., Hayes, P., McCusker, J.P., McGuinness, D., Thompson, H.S.: When owl:sameAs isn't the same: An analysis of identity in linked data. In *Proc. ISWC*, pages 305–320, 2010.

[46] He, B., Patel, M., Zhang, Z., Chang, K.C.: Accessing the deep Web: A survey. *Communications of the ACM*, 50(2):94–101, 2007.

[47] Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL Class Descriptions on Very Large Knowledge Bases. *Int. J. Semantic Web Inf. Syst.*, 5(2), 2009.

[48] A. Hogan.: Performing object consolidation on the semantic Web data graph. In *Proc. I3*, 2007.

[49] Hogan, A., Polleres, A., Umbrich, J., Zimmermann, A.: Some entities are more equal than others: statistical methods to consolidate linked data. In *Proc. NeFoRS*, 2010.

[50] Horrocks, I., Patel-Schneider, P.: Reducing OWL entailment to description logic satisfiability. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):345–357, 2004.

[51] Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic Web. In *Proc. WWW*, pages 87–96, 2011.

[52] Hu, w., Chen, J., Zhang, H., Qu, Y.: How matchable are four thousand ontologies on the semantic Web. In *Proc. ESWC*, pages 290–304, 2011.

[53] Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proc. ICDM*, 2003.

[54] Inokuchi, A., Washio, T., Motoda, H.: A general framework for mining frequent subgraphs from labeled graphs. *Fundamenta Informaticae*, 66(1-2):53–82, 2005.

[55] Ipeirotis, P.G., Gravano, L.: Distributed search over the hidden Web: Hierarchical database sampling and selection. In *Proc. VLDB*, pp. 394–405, 2002.

[56] Isaac, A., van der Meij, L., Schlobach, S., Wang, S.: An empirical study of instance-based ontology matching. In *Proc. ISWC*, pages 253–266, 2007.

[57] Jean-Mary, Y., Shironoshita, E., Kabuka, M.: Ontology matching with semantic verification. *J. Web Semantics*, 7(3):235–251, 2009.

[58] Jiang, T., Tan, A.H., Wang, K.: Mining Generalized Associations of Semantic Relations from Textual Web Content. *IEEE Trans. Knowl. Data Eng.*, 19(2), 2007.

[59] Jozefowska, J., Lawrynowicz, A., Lukaszewski, T.: The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory Pract. Log. Program.*, 10(3), 2010.

[60] Kuramochi, M., Karypis, G.: Frequent Subgraph Discovery. In *ICDM '01*. IEEE Computer Society, 2001.

[61] Kutty, S., Nayak, R.: Clustering xml documents using closed frequent subtrees — a structural similarity approach. In *5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX*, pp. 183–194, 2008.

[62] Kutty, S., Nayak, R.: Frequent pattern mining on xml documents. Chapter 14 In *Handbook of Research on Text and Web Mining Technologies*, pp. 227-248, Idea Group Inc., USA, 2008.

[63] Landauer, T.K., Foltz, P.N., Laham, D.: An introduction to latent semantic analysis. *Discourse Processes*, (25):259–284, 1998.

[64] Lee, D., Sebastian Seung, H.: Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pp. 556–562, 2000.

[65] Lehmann, J.: DL-Learner: Learning Concepts in Description Logics. *Journal of Machine Learning Research (JMLR)*, 10, 2009.

[66] Li, J., Tang, J., Li, Y., and Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. *IEEE TKDE*, 21(8):1218–1232, 2009.

[67] Madhavan, J., Halevy, A.Y., Cohen, S., Dong, X., Jeffery, S.R., Ko, D., Yu, C.: Structured data meets the Web: A few observations. *IEEE Data Engineering Bulletin*, 29(4):19–26, 2006.

[68] Maedche, A., Staab, S.: Discovering Conceptual Relations from Text. In *ECAI'00*, 2000.

[69] Maedche, A., Zacharias, V.: Clustering Ontology-Based Metadata in the Semantic Web. In *PKDD '02*, 2002.

[70] McGuinness, D.L., Fikes, R., Rice, J., Wilder, S.: An Environment for Merging and Testing Large Ontologies. In *KR2000*, 2000.

[71] Muggleton, S.: Inverse entailment and progol. *New Generation Comput.*, 13(3&4), 1995.

[72] Murray-Rust, P., Rzepa, H.S.: Chemical Markup, XML, and the Worldwide Web Basic Principles. *Journal of Chemical Informatics and Computer Science*, 39:928–942, 1999.

[73] Nayak, R.: Fast and effective clustering of XML data using structural information. *Knowledge and Information Systems*, 14(2):197–215, 2008.

[74] Nayak, R.: XML data mining: Process and applications. Chapter 15 In *Handbook of Research on Text and Web Mining Technologies*, pp. 249 - 272, Idea Group Inc., USA, 2008.

[75] Nayak, R., Iryadi, W.: XML schema clustering with semantic and hierarchical similarity measures. *Knowledge-based Systems*, 20:336–349, 2007.

[76] Nayak, R., Tran., T: A progressive clustering algorithm to group the XML data by structural and semantic similarity. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(4):723–743, 2007.

[77] Nayak, R., and Zaki, M.J. *Knowledge Discovery from XML documents: PAKDD 2006 Workshop Proceedings*, volume 3915, 2006.

[78] Network Working Group. Uniform Resource Identifier (URI): Generic Syntax, 2005. `http://tools.ietf.org/html/rfc3986`.

[79] Nebot, V., Berlanga, R.: Finding association rules in semantic web data. *Knowl.-Based Syst.*, 25(1), 2012.

[80] Nebot, V., Llavorim, R.B.: Mining Association Rules from Semantic Web Data. In *IEA/AIE (2)*, 2010.

[81] Noessner, J., Niepert, M., Meilicke, C., and Stuckenschmidt, H.: Leveraging terminological structure for object reconciliation. In *Proc. ESWC*, pages 334–348, 2010.

[82] Noy, N.F., Musen, M.A.: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI/IAAI '00*. AAAI Press, 2000.

[83] Raghavan, S., Garcia-Molina, H.: Crawling the hidden Web. In *Proc. VLDB*, pp. 129–138, 2001.

[84] Ru, Y., Horowitz, E.: Indexing the invisible web: a survey. *Online Information Review*, 29(3):249–265, 2009.

[85] Saïs, F., Pernelle, N., Rousset, M.C.: L2R: A logical method for reference reconciliation. In *Proc. AAAI*, pages 329–334, 2007.

[86] Saïs, F., Pernelle, N., Rousset, M.C.: Combining a logical and a numerical method for data reconciliation. *J. Data Semantics*, 12:66–94, 2009.

[87] Salton, G., McGill, M.J.: *Introduction to Modern information Retrieval.* McGraw-Hill, 1983.

[88] Schoenmackers, S., Etzioni, O., Weld, D.S., Davis, J.: Learning first-order Horn clauses from web text. In *EMNLP '10*. Association for Computational Linguistics, 2010.

[89] Senellart, P.: *Comprendre le Web caché. Understanding the Hidden Web.* PhD thesis, Université Paris XI, Orsay, France, December 2007.

[90] Senellart, P., Mittal, A., Muschick, D., Gilleron, R., Tommasi, M.: Automatic wrapper induction from hidden-Web sources with domain knowledge. In *WIDM*, pp. 9–16, 2008.

[91] Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3):157–168, 2011.

[92] Suchanek, F.M., Kasneci, G., Weikum, G: YAGO: A core of semantic knowledge. Unifying WordNet and Wikipedia. In *Proc. WWW*, pp. 697–706, 2007.

[93] Talukdar, P.P., Wijaya, D., Mitchell, T.: Acquiring temporal constraints between relations. In *CIKM'12*, 2012.

[94] Taylor, P. and Isard A.: SSML: A Speech Synthesis Markup Language. *Speech Communication*, 21(1-2):123–133, 1997.

[95] Tran, T., Nayak, R., Bruza, P.: Combining structure and content similarities for xml document clustering. In *AusDM*, pp. 219–226, 2008.

[96] Totten, G., Bates, C., Clinton, N.: *Handbook of Quench Technology and Quenchants.* ASM International, 1993.

[97] Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, R.: Sig.ma: live views on the web of data. In *Proc. WWW*, pp. 1301–1304, 2010.

[98] Udrea, O., Getoor, L., Miller, R.J.: Leveraging data and structure in ontology integration. In *Proc. SIGMOD*, pp. 449–460, 2007.

[99] Varde, A., Begley, E., Fahrenholz, S.: MatML: XML for Information Exchange with Materials Property Data *Proc. ACM KDD DM-SSP Workshop*, 2006.

[100] Varde, A., Maniruzzaman, M., Sisson Jr., R.: QuenchML: A Semantics-Preserving Markup Language for Knowledge Representation in Quenching. *AIEDAM Journal*, Cambridge University Press, Volume 27, pp. 65–82, 2012.

[101] Varde, A., Rundensteiner, E., Mani, M., Maniruzzaman, M., Sisson Jr., R.: Augmenting MatML with Heat Treating Semantics. *ASM International's Symposium on Web-Based Materials Property Databases*, 2004.

[102] Varde A., Rundensteiner, E., Fahrenholz S.: XML Based Markup Languages for Specific Domains. Book Chapter to appear in *Web Based Support Systems*, Springer-Verlag, UK, pp. 215-238, 2010.

[103] Villaverde, J., Persson, A., Godoy, D., Amandi, A.: Supporting the discovery and labeling of non-taxonomic relationships in ontology learning. *Expert Systems with Applications*, 36(7), 2009.

[104] Völker, J., Niepert, M.: Statistical schema induction. In *ESWC'11*, 2011.

[105] Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the Web of data. In *Proc. ISWC*, pp. 650–665, 2009.

[106] Wang, S., Englebienne, G., Schlobach, S.: Learning concept mappings from instance similarity. In *Proc. ISWC*, pp. 339–355, 2008.

[107] World Wide Web Consortium. W3C Semantic Web Activity, 1994. `http://www.w3.org/2001/sw/`

[108] World Wide Web Consortium. RDF Primer (W3C Recommendation 2004-02-10). http://www.w3.org/TR/rdf-primer/, 2004.

[109] World Wide Web Consortium. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 2004-02-10.

[110] RDF/XML Syntax Specification (Revised), W3C Recommendation, 2004. http://www.w3.org/TR/rdf-syntax-grammar/

[111] World Wide Web Consortium. SPARQL Query Language for RDF (W3C Recommendation 2008-01-15). http://www.w3.org/TR/rdf-sparql-query/.

[112] W3C. Web Services Glossary, February 2004. http://www.w3.org/TR/ws-gloss/

[113] Wu, W. Doan, A., Yu, C.T., Meng, W.: Bootstrapping domain ontology for semantic Web services from source Web sites. In *Technologies for E-Services*, pp. 11–22, 2005.

[114] World Wide Web Consortium. XML Schema Part 2: Datatypes Second Edition, 2004. http://www.w3.org/TR/xmlschema-2/

[115] Yokota, K., Kunishima, T., Liu, B.: Semantic Extensions of XML for Advanced Applications. *IEEE Australian Computer Science Communications Workshop on Information Technology for Virtual Enterprises*, 23(6):49-57, 2001.

[116] Zaki, M.J.: Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2005, 17(8):1021–1035, 2005.

[117] Zhong, C., Bakshi A. and Prasanna, V.: ModelML: a Markup Language for Automatic Model Synthesis. *IEEE International Conference on Information Reuse and Integration* pp. 317–342, 2007.