The Knowledge Engineering Review, Vol. 00:0, 1–30. © 2013, Cambridge University Press
DOI: 10.1017/S0000000000000000. **NOTICE: this is the authors version of a work that was accepted for publication. Changes resulting from the publishing process may not be reflected in this document.** Printed in the United Kingdom

XML Clustering: A Review of Structural Approaches

Maciej Piernik, Dariusz Brzezinski, Tadeusz Morzy and Anna Lesniewska

Institute of Computing Science, Poznan University of Technology, 60-965 Poznan, Poland
E-mail: mpiernik, dbrzezinski, tmorzy, alesniewska@cs.put.poznan.pl

Abstract

With its presence in data integration, chemistry, biological and geographic systems, XML has become an important standard not only in computer science. A common problem among the mentioned applications involves structural clustering of XML documents — an issue that has been thoroughly studied and led to the creation of a myriad of approaches. In this paper, we present a comprehensive review of structural XML clustering. First, we provide a basic introduction to the problem and highlight the main challenges in this research area. Subsequently, we divide the problem into three subtasks and discuss the most common document representations, structural similarity measures, and clustering algorithms. Additionally, we present the most popular evaluation measures, which can be used to estimate clustering quality. Finally, we analyze and compare 23 state-of-the-art approaches and arrange them in an original taxonomy. By providing an up-to-date analysis of existing structural XML clustering algorithms, we hope to showcase methods suitable for current applications and draw lines of future research.

1 Introduction

Since its introduction in 1996, the eXtensible Markup Language (XML) has become a standard for developing web applications dealing with document storage and retrieval. As a result, a lot of work has been done in the field of XML document processing and management. In the past few years, the issue of XML document mining has gained a lot of attention. One of the most interesting and intensively investigated fields in this research area is XML document clustering.

In general, clustering is a process aiming at grouping together similar objects. Clustering techniques for traditional, textual documents have been developed for many years (Aggarwal & Zhai 2012). However, these techniques are usually inappropriate for clustering of XML documents (Dalamagas et al. 2004). The key characteristic that distinguishes XML documents from traditional ones is their semistructured nature. An XML document consists of structure — formed by tags and relationships between them, and content — the actual data stored in the document. Therefore, there is a need to develop new clustering algorithms specifically designed for XML documents. This need has led to the proposal of several methods, which can be classified according to: structure/content usage, level of analysis, document representation, and cohesion of data sources. The following paragraphs discuss features taken into account in all of the mentioned classifications.

Concerning structure/content usage, XML clustering methods can be categorized into three main groups. The first group treats an XML document as plain text (Aggarwal & Zhai 2012), the second group utilizes both content and structure (Tagarelli & Greco 2010, Kutty et al. 2008, 2010, 2011, Doucet & Lehtonen 2006), while the third group omits the content of a document and relies solely on its structure. Content-only methods, which aim at creating textually similar groups of documents, utilize existing text mining algorithms. On the other hand, structural clustering methods require XML-specific data representation and processing, and, therefore, cannot be easily generalized from traditional clustering algorithms (Zaki & Aggarwal 2006). Nevertheless, structural analysis has proved particularly useful in scenarios involving large and complex datasets. In such cases, analysis of content and semantics of documents often becomes impossible while structural analysis remains feasible. In this review, we will focus solely on structural approaches.[1]

Structural analysis of XML can be performed based on two levels: document structure and document metadata. If no document metadata is available, one can study document trees, tags, edges, and structure-derived features, e.g., quantity of nodes, height or width of a document tree. On the other hand, the structure of every XML can be described by either a Document Type Definition (DTD) or an XML Schema (XSD). If such document descriptions are available, one should decide on whether to deduct structural similarity from DTDs and XSDs (Bertino et al. 2008, Li et al. 2007) or rely only on document structures. During the description of existing XML clustering algorithms we will focus on the schemaless approaches.

Apart from determining the level of the analysis, one has to select a document representation. Various representations of XML structure, such as graphs, trees, path sets, tag vectors, or time series have been proposed. The more complex the representation, the more accurate the analysis. However, increasing the complexity of representation leads to more elaborate and time-consuming solutions in further stages of clustering. The chosen document representation partially defines available similarity measures and clustering approaches and, therefore, is one of the most important steps in defining a clustering algorithm.

Finally, when choosing a method for clustering XML documents by structure, one of the key tasks is to identify the character of a data source. In this aspect, two cases can be considered: documents originating from the same source or from different sources, i.e., homogeneous or heterogeneous data sources. This distinction has a major influence on the problem's complexity. Documents originating from heterogeneous data sources are generally less difficult to cluster due to easily identifiable differences in tag labels. In such cases, lightweight document representations, such as tag or edge vectors, are likely to be sufficient. However, the analysis conducted on documents originating from heterogeneous data sources often faces the problem of nomenclature ambiguity (e.g., the same tag names may be written in different ways or elements may be placed in different order), which can be misleading when simple representations are utilized. Homogeneous documents, on the other hand, often share the same tag vocabulary. In this case, more complex representations, like paths or trees, are appropriate, as they allow for the use of more sophisticated similarity measures. In general, finding appropriate functions for determining the similarity of documents is a difficult task.

Taking into account the rapid development of structural XML clustering in recent years, there is a need for a comprehensive, up-to-date review of the research performed so far to unify the concepts and terminology among the researchers and to survey the state-of-the-art methodologies investigated over the past.

Several reviews related to XML clustering are available. However, they either do not focus exclusively on structural approaches or relate to specific parts of the clustering process. Thus, these reviews are only partially related to structural clustering and are mostly outdated. Currently, the most cited survey related to XML clustering by structure was published back in 2004 (Buttler 2004) and concentrated solely on similarity computation, rather than the entire clustering process. Furthermore, the author reviews only some of the now available structural measures without providing a general overview or taxonomy. Other reviews limited to similarity computation include (Guerrini et al. 2007), where eight measures are discussed, not all of which are XML-specific, and (Tekli et al. 2009) which concentrates mainly on tree-edit distance measures also for schema comparison and content-based information retrieval. The only survey covering the entire XML clustering process (Algergawy et al. 2011), although comprehensive,

[1]For a systematic review of textual clustering methods and clustering in general the reader is referred to (Aggarwal & Zhai 2012, Jain et al. 1999, Xu & Wunsch 2005)

does not concentrate solely on structure-based methods. Moreover, since its publication many new algorithms appeared in addition to the 9 structural approaches presented in the cited review. Finally, several reviews are limited to specific application fields. For example, (Vakali et al. 2007) and (Husek et al. 2007) focus on general clustering methods for Web documents.

The aim of this paper is to provide an extensive, up-to-date overview of XML document clustering by structure. We will give the motivation behind structural clustering and present various real-world applications for this task. Furthermore, we will analyze each step of the clustering process and survey 23 state-of-the-art structural XML clustering methods. Finally, we will highlight the main open issues and indicate lines of future research. By presenting a comprehensive review of existing approaches we hope to provide a valuable resource for researchers and developers seeking state-of-the-art XML clustering algorithms.

The remainder of the paper is organized as follows. Section 2 describes motivation for XML clustering by structure along with some of its applications. In Section 3, we present a general framework for XML document clustering. Sections 4–7 describe the main steps of the framework in detail, i.e., document representation, similarity measures, clustering, and evaluation, respectively. Next, Section 8 gives an overview of existing XML document clustering algorithms. Section 9 highlights open issues and discusses future research directions in the field of XML clustering by structure. Finally, Section 10 summarizes the survey.

2 Motivation and applications

The popularity of XML has led to its adoption in many different domains outside information technology, like medicine (HL7 CDA (Dolin et al. 2006)), mathematics (MathML (*Mathematical Markup Language - MathML* 1998)), chemistry (CML (*Chemical Markup Language - CML* 1995)), or biology (PDBML (Westbrook et al. 2005)). Therefore, many data-oriented applications, which process semistructured documents, could benefit from dedicated XML clustering algorithms. Example clustering tasks in such applications include: finding groups of customers with similar behavior, personalizing content delivery from news feeds, comparing chemical compounds and biochemical structures, or clustering observed earthquake epicenters to identify dangerous zones (Crescenzi et al. 2001, 2005, Sankoff & Kruskal 2000, Somervuo & Kohonen 2000, Wilson et al. 2003, Zhu et al. 2010, Zheng et al. 2011).

```
A1:
<article>
 <title>XML</title>
 <author country="US">
  <first>John</first>
  <last>Doe</last>
 </author>
 <image>jpg</image>
</article>
```

```
A2:
<article>
 <title>Medicine</title>
 <author>
  <first>Henry</first>
  <last>Smith</last>
 </author>
 <image>bmp</image>
</article>
```

```
A3:
<news>
 <title>Curiosity</title>
 <pict>surface</pict>
 <author>
  <name>Anna</name>
 </author>
<body>Landing</body>
</news>
```

```
A4:
<news>
 <title>Mars</title>
 <author>
  <name>Ronda</name>
 </autor>
 <sum>23</sum>
 <body>Total</body>
</news>
```

Figure 1 Examples of XML documents

To illustrate the importance of structural clustering, let us analyze a practical example. Figure 1 presents 4 XML documents from different content providers. Let us assume a user of the content delivery system that provided these documents wishes to automatically categorize articles to facilitate future reading. By applying a simple structural XML clustering algorithm, e.g., one analyzing document tag counts, the documents could form two groups: one containing articles A1 and A2, and the other containing articles A3 and A4. With the documents clustered, the user may prioritize certain document groups while ignoring others. The use of a structural clustering method in this scenario is a practical choice as content providers may offer hundreds of articles each spanning many pages. Therefore, an application using a textual clustering method would have to analyze thousands of words and require substantial processing time, while an application using a structural method could quickly and accurately distinguish providers and, possibly, article topics.

Information filtering is merely one of many practical problems which may benefit from structural clustering. Below, we discuss some of the most common applications of grouping large datasets of semistructured documents.

Web mining

One of the biggest sources of XML documents is the Internet. The Web has become a distributed data and service repository which holds such vast amounts of information that it requires dedicated solutions for management and processing. Structural clustering enables effective detection of similarities between documents gathered on the Internet. This information can be applied to solve a variety of Web mining problems, such as document source recognition or structural analysis of websites. XML clustering can also be used to form groups of similar Web pages which later can serve as Web wrappers, i.e., programs that extract data from HTML pages, and transform them into a machine-processable format. By combining structural clustering and Web wrappers, financial data published by several specialized Web sites only in HTML can be constantly extracted and processed for mining purposes. Similarly, data delivered on the Web by thematic communities can be extracted and integrated via structural XML clustering (Crescenzi et al. 2001, 2005).

XML data integration

XML is prevalent among standards for exchanging and integrating data between Web sites as well as other data-oriented applications. One of the most significant steps of data integration is identifying structurally and semantically similar documents (Viyanon et al. 2008, Lee et al. 2002). Such an analysis can be performed to find the same pieces of information presented in different forms or to identify human errors which occurred during the creation of documents.

Bioinformatics

Natural sciences, such as biology and chemistry, use the XML standard as a means of representing hierarchies and relationships. As XML can be easily exchanged among several research groups, it has been used in collaborative environments for pedigree data management (Achard et al. 2001). Another example of structural clustering in biology is the discovery of homologous proteins, i.e., sets of proteins sharing similar structures. Other applications in this field include gene clustering (Andreopoulos et al. 2009) and DNA/protein sequence clustering (Sankoff & Kruskal 2000, Somervuo & Kohonen 2000). The use of structural algorithms in these areas is especially beneficial as real-world biological datasets are extremely large and require light-weight processing.

Spatial data management

The hierarchical nature of XML is often used to represent documents describing spatial data. XML grammars, such as the Geographical Markup Language (GML (Pospech 2009)) or the Keyhole Markup Language (KML (Wernecke 2008)), are used as modeling languages and Web service interchange formats. In these languages, spatial information like areas that include lakes, rivers, wells, or farms can be represented as tree structures. Furthermore, XML is also used to describe waypoints, tracks, and routes on maps. Structural clustering of spatial objects and routes can help to identify similar geographic objects, e.g., large areas with similar lake-forest spatial arrangements (Wilson et al. 2003, Zhu et al. 2010) or tourist travel patterns (Zheng et al. 2011).

In all of the above scenarios, XML clustering by structure plays a crucial role in processing large volumes of data. It is worth noting that, although the described applications differ in the expected outcome and objective, the clustering process itself remains the same and can be formulated into a general XML clustering methodology.

3 Clustering methodology

Cluster analysis can be divided into three basic steps, forming a general clustering methodology (Jain et al. 1999, Algergawy et al. 2011), presented in Figure 2.

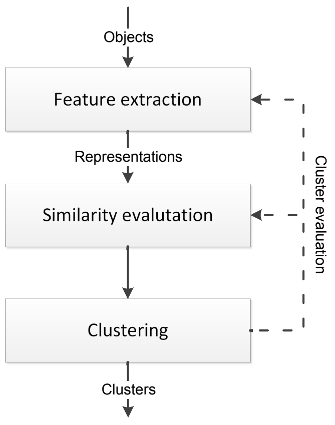


Figure 2 Clustering methodology

Feature extraction
Before input objects can be grouped or even compared, they have to be transformed into a representation that allows for effective processing. If one is to cluster news articles from different content providers, he/she has to extract essential information from those articles and represent it in a form that will facilitate article comparison. In the field of XML clustering, this step is especially important as XML documents contain not only textual but also structural information and can grow fairly large in size. Therefore, the developers of XML clustering algorithms should aim at using descriptive yet compact representations. The most popular XML document representations include graphs, trees, vectors, and sets of paths. These and more possible representations will be discussed in detail in Section 4.

Similarity evaluation
Once the documents are represented in a concise format they can be compared according to a chosen similarity measure. If the first step of the presented clustering methodology was meant to extract interesting features from input objects, then the aim of the second step is to determine which of the selected features, and to what extent, decide whether two objects are similar. Since we can only use the features available in the transformed objects, it becomes apparent that the selected object representation has a huge impact on similarity evaluation. Following our example, depending on the representation we selected to encode news articles, we can choose to calculate their similarity according to section titles, keywords, section-subsection hierarchy, or even the number of paragraphs. Typical XML structural similarity measures, including tree-edit distance, vector distance, and other measures, will be discussed in Section 5.

Clustering
After comparing the input objects we can use the obtained knowledge to group them into clusters. The information about which pairs of objects are alike is usually stored in a similarity matrix, which contains data describing the distances between all documents in the dataset. There are many clustering algorithms which use the results of similarity evaluation to group documents. For the purposes of this paper we divide the clustering algorithms into two main groups of methods: hierarchical and flat. The most popular examples of these approaches in the context of XML clustering will be discussed in Section 6.

Since clustering is a form of unsupervised learning from data, it requires automatic evaluation methods rather than supervised training. For this reason, many clustering methods restart the analysis after evaluating the obtained results — a situation depicted with a feedback loop in Figure 2. In this case, clustering algorithms use evaluation methods, also called *validation indices*, which consider factors like intra-cluster document similarity, dissimilarity of documents from different clusters, or coverage of the dataset. Validation indices are required to ensure that obtained clusters are of practical value and are especially needed when dealing with high-dimensional data, where clusters cannot be easily visualized and verified. Regarding the example of clustering news articles, depending on the selection of a clustering algorithm and evaluation measure, the user can obtain groups of articles which are very dissimilar between groups or, in contrast, groups that overlap. Section 7 discusses the most popular evaluation methods used in XML clustering.

In the following sections, we discuss common approaches to each step of the presented clustering methodology. Although the approaches will be discussed separately for each step, it should be remembered that in order to create a successful clustering algorithm the selected methods should be carefully matched. As XML is popular among many domain specific applications, structural clustering algorithms work best when tailored to the needs of the end-user. Therefore, no single combination of the presented approaches is best for all applications and end-requirements should be taken into account when designing new XML clustering algorithms.

4 Representations

The first step of the clustering methodology involves transforming objects into a chosen representation. As we are analyzing XML documents, let us first present the basic concepts related to the XML format. An XML document consists of *elements* that are textual data structured by *tags*. Each element consists of a start and end tag, optional *attributes* defined as key-value pairs, and elements or textual data between the tags. For example, in article A1 in Figure 1, element `author` consists of an attribute `country` with value "US" and two sub-elements `first` and `last`, which contain textual data "John" and "Doe", respectively.

Many approaches to representing XML structure have been proposed, however, the most commonly adopted ones are trees and vectors. Other interesting document representations include: matrices, time series, and single numerical values describing certain document features (e.g., number of distinct tags, width or height of a document tree). The following subsections discuss the most common XML document representations in detail. Each of the presented representations will be illustrated with an example based on a sample XML document D1 listed in Figure 3.

```
<articles>
    <article>
        <title>
            <article/>
        </title>
        <author/>
        <reference>
            <article>
                <title>
                    <article/>
                </title>
            </article>
        </reference>
    </article>
</articles>
```

Figure 3 Example XML document D1

4.1 Tree

One of the most common representations of an XML document is a rooted labeled tree defined as a 4-tuple $T = (N_T, E_T, L_T, \alpha)$, where:

- $N_T = \{n_{root}, n_2, ..., n_n\}$ is a finite set of uniquely identified nodes, where n_{root} represents the root node;
- $E_T = \{(n_i, n_j) : n_i, n_j \in N_T\}$ is a finite set of edges, where n_i is the parent node of n_j;
- L_T is a set of node labels corresponding to element and attribute names from an XML document;
- $\alpha : N_T \to L_T$ is a function mapping each node into a label.

This representation is often further restricted with a left-to-right order among siblings. When such order is provided, the tree is called *ordered.* Figure 4 illustrates the structure of an XML document represented as a rooted ordered labeled tree.

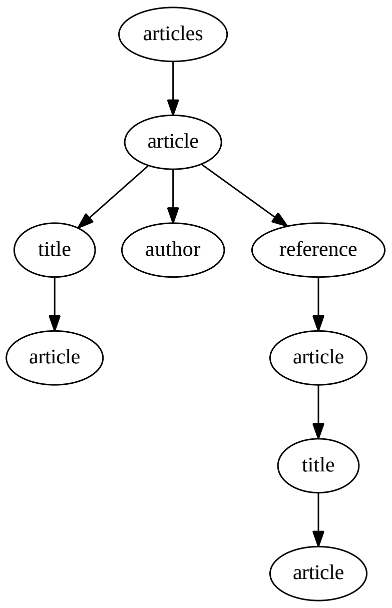


Figure 4 Tree representation

In an XML document tree, we can distinguish several relationships among the elements:

- *parent-child*: a relationship between each element node and its direct subelement/attribute;
- *ancestor-descendant*: a relationship between each element node and its direct or indirect subelement/attribute;
- *order*: a relationship between siblings.

It is worth mentioning that, although there are in fact two node types (elements and attributes), in the vast majority of existing approaches they are treated equivalently.

4.2 Vector

The tree representation is the most natural and informative for XML documents, but often requires complex computations. Decomposing a tree into smaller parts is usually accompanied by information loss, however, it may reduce the complexity of similarity evaluation. The information loss involves reducing the number of relationships between tree nodes. Depending on the extent of reduction we can obtain *paths*, *edges*, or, after reducing all node relationships, individual *tags*.

A path is a list of consecutive nodes in a tree according to the parent-child relationship. A path beginning with a root node and ending with a leaf is called a *full path*. If we focus only on the direct relations between tree nodes, we can represent a document as a group of edges. This way the parent-child relationships remain, but the complexity reduction is very high. Finally, we can remove all relationships between the nodes, leaving only node labels.

As described above, the vector representation may reduce the structural information to a varied extent. Several approaches based on this representation have been proposed (Theobald 2003, Tran et al. 2008, Yoon et al. 2001, Lesniewska 2009) and most of them use a full path decomposition. Each XML document is modeled as a vector $\vec{v} \in \mathcal{R}^m$, where m is the number of different full paths in the dataset and each element v_i represents the frequency of a single full path in that document. Together, all document vectors produce a 2-dimensional matrix $m \times n$ representing the whole dataset, where n is the number of all documents. An example of such a representation is shown in Table 1, where D1 is the sample document from Figure 3 and D2 and D3 are other documents in the dataset.

This approach can be slightly modified by limiting the maximal length of the paths. Naturally, such a modification further reduces the structural information in the documents, however, it may aid the process of similarity computation, because shorter paths are more likely to co-occur in the compared documents. An example of a vector representation with path length limited to 1 (tag-only approach) is presented in Table 2.

Table 1 Vectors of full path frequencies

Full path	D1	D2	D3
articles/article/title/article	1	0	0
articles/article/author	1	0	0
articles/article/reference/article/title/article	1	5	0
articles/news/title	0	2	0
articles/news/number	0	2	0
news/title	0	0	1
news/author/name	0	0	1

Table 2 Vectors of label frequencies

Label	D1	D2	D3
articles	1	1	0
article	4	12	0
title	2	7	1
author	1	0	1
reference	1	5	0
news	0	2	1
number	0	2	0
name	0	0	1

4.3 Other

A structure which decomposes documents in a similar way in which the vector representation does, is the set representation. In this approach, a document is also broken down into parts, such as paths or edges, however, it is not encoded into a numerical vector. Conversely, a set (or a multiset) consisting of these parts constitutes the representation. In this way, even though some structural information is lost during the decomposition, it is still possible to directly compare the parts of the documents when evaluating their similarity. Such a comparison is impossible when documents are encoded into numerical vectors. Figure 5 shows the sample document D1 represented as a set of paths.

$p^T = \{$ articles/article/title/article,
articles/article/title,
articles/article,
articles/article/author,
articles/article/reference/article/title/article,
articles/article/reference/article/title,
articles/article/reference/article,
articles/article/reference $\}$

Figure 5 Set of paths representation

The structure of an XML document may also be transformed into a graph. An example of a graph representation is the *s-graph* (Lian et al. 2004). Given an XML document d, an s-graph of d, $sg(d) = (N, E)$, is a directed graph such that N is the set of all the elements and attributes in d and E is the set of all edges in d. An s-graph based on our sample XML document D1 is depicted in Figure 6.

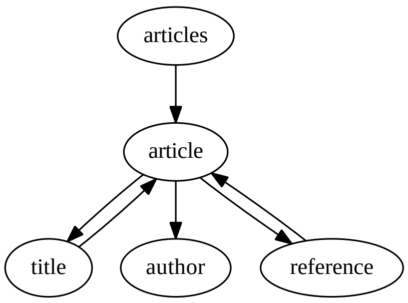
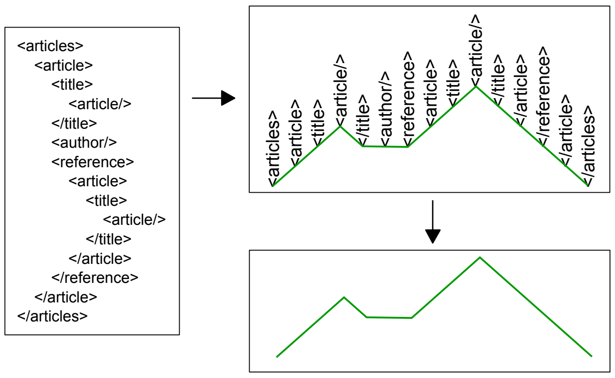


Figure 6 S-graph representation

An XML document can also be represented as a time series where each occurrence of a tag corresponds with an impulse (Flesca et al. 2005). In this case, node labels are completely omitted and only the element nesting is taken into account. To better explain this representation, one can visualize an XML document rotated by 90 degrees with a line drawn following the document's indentation (the higher the node level in a document tree the bigger the indentation). Figure 7 shows the sample XML document D1 represented as a time series.



```
<articles>
  <article>
    <title>
      <article/>
    </title>
    <author/>
    <reference>
      <article>
        <title>
          <article/>
        </title>
      </article>
    </reference>
  </article>
</articles>
```

Figure 7 Time series representation

Another approach, one of the most lightweight representations of XML, relies on transforming each document into a single numeric value, called a *feature* (Lesniewska & Primke 2008). This number represents some general structural characteristic of a document, e.g., number of nodes, number of leaves, number of distinct labels, number of distinct full paths, or maximal path length. While highly efficient, this representation significantly reduces the information collected from the documents and is insufficient for most XML clustering applications. Nevertheless, it may be used for preliminary analysis of very large datasets.

5 Similarity measures

The second step of the clustering methodology aims at computing similarity between all documents in a dataset. The set of available similarity measures is, to a large extent, dictated by the choice of document representation. The diversity of representations results in a wide variety of measures to choose from. In this section, we will focus on the most popular similarity measures corresponding with the previously presented document representations.

5.1 Tree

The most natural and widely used representation of XML is the rooted ordered labeled tree, as it fully preserves the hierarchical nature of the documents. If one intends to utilize this information for similarity evaluation, one has to use a measure which is capable of comparing tree representations. The most widely used method for computing the distance between trees is called *tree-edit distance* and is calculated as the minimal number of predefined operations required to transform one tree into another. Such a procedure computes the similarity between entire documents, not just their parts.

The tree-edit distance is associated with three atomic edit operations conducted on nodes of a rooted ordered labeled tree: deletion, insertion, and relabeling. Let t_1 and t_2 be a pair of rooted ordered labeled trees. A tree-edit sequence is a sequence of tree operations that transforms t_1 into t_2. If we assign a cost to every operation, the tree-edit distance between t_1 and t_2 will be the minimum cost of all possible tree-edit sequences that transform t_1 into t_2. Below we present selected tree-edit distance methods chosen based on their novelty in terms of allowed edit operations and applicability in XML processing.

One of the earliest approaches to evaluating similarity between trees using sequences of simple edit operations was introduced by Selkow (Selkow 1977). This solution, however, restricted the insertion and deletion operations to the leaf nodes exclusively and had an exponential complexity. The first non-exponential approach to solving this problem was introduced by Tai (Tai 1979). The author presented an algorithm of polynomial complexity $O(|t_1||t_2|depth(t_1)depth(t_2))$, where $|t_i|$ is the number of nodes and $depth(t_i)$ is the depth of the i-th tree. Furthermore, this solution lacked the restriction on edit operations, allowing for insertion and deletion of any nodes in the tree. Currently, the most efficient tree-edit distance algorithm allowing for edit operations to appear anywhere in a tree is the RTED, proposed by Pawlik and Augsten (Pawlik & Augsten 2011).

One of the first tree-edit distance measures specific to the XML format was proposed by Chawathe (Chawathe 1999). In this approach, the author reestablished the restriction on insert and delete operations to leaf nodes exclusively. He claimed, that insert and delete operations in the middle of an XML document tree are unnatural, as they are followed by the children nodes transfer and, therefore, not only the nodes change but also the relationships between them. The proposed restriction reduced the algorithm complexity to $O(|t_1||t_2|)$ without decreasing the quality of the results. Nierman and Jagadish (Nierman & Jagadish 2002) proposed a solution in which they allow for restricted insertions and deletions of whole subtrees. A subtree s can be inserted into a tree t only if the root node of s will become a child node of one of the leaf nodes in t. A subtree s can be deleted from a tree t only if the leaf nodes of s are also the leaf nodes of t. This solution presents a slightly higher complexity than Chawate's approach, however, it outperforms his algorithm in terms of clustering quality.

In Table 3 we have summarized the most important tree-edit distance approaches in the context of XML similarity. The table presents the algorithms along with their supported operations and complexities.

Table 3 Summary of most relevant approaches to tree-edit distance

Approach	Operations	Complexity	Notation
(Selkow 1977)	relabel node, insert node*, delete node*	$4^{min(\|t_1\|\|t_2\|)}$	t_i – number of nodes in the i-th tree
(Tai 1979)	relabel node, insert node, delete node	$O(\|t_1\|\|t_2\|d(t_1)d(t_2))$	t_i – number of nodes in the i-th tree; $d(t_i)$ – depth of the i-th tree
(Pawlik & Augsten 2011)	relabel node, insert node, delete node	$O(n^3)$	n – number of tree nodes
(Chawathe 1999)	relabel node, insert node*, delete node*	$O(NM)$	N, M – dimensions of the matrix that represents the edit graph
(Nierman & Jagadish 2002)	relabel node, insert node*, delete node*, insert tree, delete tree	$O(\|t_1\|\|t_2\|)$	$\|t_i\|$ – number of nodes in the i-th tree
(Dalamagas et al. 2006)	relabel node, insert node*, delete node*	$O(\|t_1\|\|t_2\|)$	$\|t_i\|$ – number of nodes in the i-th tree
* operations restricted to leaves			

5.2 Vector

The vector representation has two main advantages. Firstly, the values in the vector may correspond to a wide variety of features, like subtrees, paths, edges, tags, XPath queries, etc. Secondly, there already exist many different measures designed for computing similarity between two vectors.

Given two XML documents d_1 and d_2 represented as binary vectors, the distance between these documents may be defined as follows:

$$Dist(d_1, d_2) = \sum_{i=1}^{n} |v_1[i] - v_2[i]|,$$

where v_1 and v_2 denote documents d_1 and d_2 represented as binary vectors and n is the number of all distinct features in the dataset. This simple distance measure illustrates the number of features which are different in the compared documents, and may also be presented as a normalized similarity measure:

$$Sim(d_1, d_2) = 1 - \frac{\sum_{i=1}^{n} |v_1[i] - v_2[i]|}{\sum_{i=1}^{n} v_1[i] \oplus v_2[i]},$$

where $\oplus$ is a *bitwise or* operation performed on each pair of corresponding vector cells. This measure shows the percentage of common features shared between two documents.

If we represent two XML documents d_1, d_2 as real vectors v_1 and v_2 (e.g., by including the quantity of feature occurrences or by weighting features) we may use metric distances. The most commonly used metric for computing distances between two vectors is the *Minkowski* distance, calculated as:

$$L_p(v_1, v_2) = \sqrt[p]{\sum_{i=1}^{n} |v_1[i] - v_2[i]|^p},$$

where n is the number of all distinct features in the documents, and $p \in \mathcal{R}_+$ is a user-defined metric parameter. Thanks to this parameter, the Minkowski distance is in fact an infinite family of metrics. The best known and commonly used metrics from the Minkowski family are the *Euclidean* distance (L_2) and the *Manhattan* distance (L_1). Figure 8 presents some examples of Minkowski distances.

The distances calculated with the Euclidean metric can be easily analyzed because the subsequent values in a vector may be considered as coordinates of a point in n-dimensional

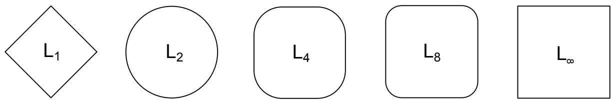


Figure 8 Examples of Minkowski distances. The figures mark an equal distance from the center point for each metric.

space, where n is the size of the vector. In the context of XML, however, where elements often appear multiple times in a single document, the *cosine distance* is a more popular choice. Instead of measuring the Euclidean distance between points in n-dimensional space, the cosine distance measures an angle between the vectors. This way, although the information about multiple label occurrences is preserved, it has a smaller impact on the result. The cosine distance between two XML documents represented as real vectors is defined as:

$$Sim_{Cos}(d_1, d_2) = \frac{v_1 \cdot v_2}{||v_1||||v_2||}.$$

Another metric used for measuring similarity between vectors is the *Tanimoto* measure. This measure is a vector version of the *Jaccard coefficient*, which is used for measuring similarity between sets. The following equation presents the definition of the Tanimoto measure:

$$Sim_{Tan}(d_1, d_2) = \frac{v_1 \cdot v_2}{||v_1||^2 + ||v_2||^2 - v_1 \cdot v_2}.$$

5.3 Other

The tree and vector representations are the most common and widely used in the XML domain. However, as presented in Section 4, many other interesting approaches exist, which also require specific measures for similarity evaluation. For instance, given two XML documents d_1 and d_2 represented as sets s^{T_1} and s^{T_2}, the aforementioned Jaccard coefficient may be used:

$$Sim_{Jac}(d_1, d_2) = \frac{|s^{T_1} \bigcap s^{T_2}|}{|s^{T_1} \bigcup s^{T_2}|}.$$

Another measure, similar to the Jaccard coefficient, was proposed to compute distances between graphs (Lian et al. 2004). Before calculating this measure, the compared documents are first transformed into s-graphs (sg_i), described earlier in Section 4.3. Next, during the similarity computation each graph is treated as the set of edges it contains. Thus, the similarity measure is defined as follows:

$$Sim_{SG}(d_1, d_2) = \frac{|sg_1 \bigcap sg_2|}{max\{|sg_1|, |sg_2|\}}.$$

When two XML documents are represented as time series, the similarity computation requires additional operations. Firstly, the documents can have different lengths, therefore, the time series may require stretching, shrinking, cutting, or other adjustments. Additionally, the corresponding signals can be shifted, thus, in order to acquire an accurate comparison, further adjustments may be necessary. These concerns led to the adoption of the discrete Fourier transform (DFT), which addresses the presented issues (Flesca et al. 2002). A distance measure based on this notion uses an interpolation of DFT ($D\hat{F}T$) with respect to the frequencies appearing in both documents, and is defined as follows:

$$Dist_{DFT}(d_1, d_2) = \sum_{k=1}^{M/2} (|[D\hat{F}T(h_1)](k)| - |[D\hat{F}T(h_2)](k)|)^2,$$

where M is the number of tags appearing in both documents, and h_i is the i-th document encoded as a time series.

One of the most efficient approaches to computing similarity between XML documents is based on entropy (Helmer 2007). This solution originates from the *normalized information distance* (*NID*), which utilizes the notion of the *Kolmogorov complexity.* The Kolmogorov complexity $K(o)$ of an object o measures the size of a minimal program required to obtain this object. The normalized information distance between objects o_1 and o_2 is defined as follows:

$$NID(o_1, o_2) = \frac{max\{K(o_1|o_2), K(o_2|o_1)\}}{max\{K(o_1), K(o_2)\}},$$

where $K(o_1|o_2)$ is the minimal size of a program producing object o_1 with the input given as object o_2.

The Kolmogorov complexity is a theoretical concept, therefore, the Normalized Information Distance cannot be used to compute similarity between objects. However, algorithmic complexity can be approximated with data compression algorithms. In such an approach, documents are compressed in a lossless manner and saved as files. The lengths of these files are later compared in order to obtain the *Normalized Compression Distance* (*NCD*). Given $C(d)$ as a compressed XML document d and $C(d_1 d_2)$ as a compressed concatenation of two documents d_1 and d_2, NCD is defined as follows:

$$NCD(d_1, d_2) = \frac{C(d_1 d_2) - min\{C(d_1), C(d_2)\}}{max\{C(d_1), C(d_2)\}}.$$

As described in Section 3, in the last step of the clustering methodology documents are grouped according to their similarity. In the next section, we present some of the most important clustering techniques which can be used in the final step of the clustering methodology.

6 Clustering techniques

In the third step of the clustering methodology, documents are grouped according to their similarity. This process may be conducted according to one of many algorithms. For the purposes of this survey we divide these methods into two groups: hierarchical and flat. Hierarchical algorithms are executed sequentially and lead to the construction of nested hierarchies of clusters, whereas flat methods form a single set of output clusters. Both approaches aim at creating a dataset partitioning which maximizes the similarity between documents in each cluster and the distance between documents from different clusters.

6.1 Hierarchical clustering

Hierarchical clustering algorithms are divided into two main families of methods: agglomerative (bottom-up) and divisive (top-down). These methods rely on iterative merging/splitting of single clusters until an algorithm-specific stop condition is reached. The result of this process is a tree chart called a dendrogram, which illustrates the order in which clusters were merged/split. It also shows to what extent the clusters are related to one another.

The first advantage of hierarchical clustering methods is their intuitiveness. They are easy to implement and easy to follow during the execution. The main advantage, however, lies in the output dendrogram as it enables researchers to analyze the relationships between the clusters as well as the order in which they were created. Furthermore, dendrograms facilitate automatic detection of the optimal number of output clusters. This can be achieved by collecting the distances between the clusters divided/merged in each step and computing their standard deviation σ. The number of output clusters may be defined by the first distance exceeding the 3σ threshold. Another benefit of hierarchical approaches is the fact that they are to some extent immune to imbalanced datasets and oddly shaped (non-spherical) clusters.

Despite many advantages and the intuitiveness of the method, hierarchical clustering has also a few drawbacks. The most important drawback is that the assignment of an object to a cluster is final, therefore, once the decision is made it cannot be changed. Another drawback originates from the iterative nature of this method and it is the algorithm's quadratic complexity, which is high compared with other approaches. The final disadvantage of hierarchical methods is their lack of a global goal function. This property, together with the fact that the object-to-cluster assignment is final, increases the danger of falling into local optimum.

The most popular method for hierarchical clustering is the *Agglomerative Hierarchical Clustering* algorithm (AHC) (Johnson 1967), which iteratively joins pairs of clusters based on their similarity. The similarity of two clusters is evaluated based on the distances between their documents. The simplicity of this algorithm and the fact that it can cope with any type of data makes it a very popular choice for XML clustering.

Another well-known method for hierarchical clustering is *CURE* (Guha et al. 1998). This algorithm partitions the dataset into a fixed number of initial clusters and later agglomerates them in a hierarchical fashion, shifting the centroids of the initial clusters towards the centroids of the agglomerated clusters. This mixture of hierarchical and partition-based approaches assures lower complexity than the AHC algorithm. Furthermore, it copes well with imbalanced data, non-spherical clusters and outliers. However, studies have shown that the parameter setting has a significant influence on the results (Han 2005), so in order to assure high clustering quality, the values of the parameters have to be carefully selected. Moreover, this method is not applicable to categorical data. That is why, in the context of XML clustering a variation of the CURE algorithm is used, called *ROCK* (Guha et al. 2000), which is capable of dealing with categorical data.

6.2 Flat clustering

The second family of clustering approaches is constituted by flat methods. In these approaches, unlike the hierarchical methods, in subsequent steps of algorithm execution objects can be relocated between clusters until reaching an algorithm-specific stop condition. The ability to relocate objects between clusters is one of the biggest benefits of flat approaches. Another advantage of flat clustering over hierarchical approaches is lower complexity, because the number of iterations is dictated by the desired accuracy, not the number of objects in the dataset. The main disadvantage of non-hierarchical clustering methods is that they usually begin with a random partition, which is only further improved. This leads to a risk of falling into a local optimum as the algorithm tries to improve the partition in each consecutive step.

The most important representative of the flat approaches is the *k-means* algorithm (Hartigan & Wong 1979). The algorithm forms k initial clusters around randomly chosen points and iteratively improves them. The main drawback of this approach is that centroids are calculated based on the positions of all objects in each cluster. As a result, this method is highly sensitive to outliers. Moreover, due to the fact that objects are assigned to clusters based on their centroids, k-means does not cope well with non-spherical clusters and clusters with different densities. Nevertheless, this method is still a popular choice in the context of XML clustering.

The problem of spherically shaped clusters is eliminated in density-based algorithms. In these approaches, clusters are formed based on the density of objects in a given neighborhood, rather than their distances to cluster means. Additionally, these methods usually require only a single full dataset scan. The most important representatives of density-based clustering algorithms are *DBSCAN* (Ester et al. 1996) and *OPTICS* (Ankerst et al. 1999). Among other popular flat clustering algorithms which are used in the context of XML are: EM (Dempster et al. 1977) — useful with imbalanced datasets, and CLOPE (Yang et al. 2002) — useful when treating XML documents as transactions (e.g., binary vector of frequent items may be considered as a transaction).

7 Evaluation methods

After cluster analysis, the resulting clusters should be validated in order to verify if they are of acceptable quality and if further clustering needs to be performed. Such an evaluation can be carried out by using either an internal or external measure (Tan et al. 2005).

Internal measures (also called *internal indices* or *unsupervised measures*) measure the quality of clusters without the use of any external information about the way in which the analyzed objects should be clustered. Unsupervised measures are divided into measures of cluster *cohesion*, which determine the compactness of objects within a cluster, and cluster *isolation*, which determine how well a cluster is separated from other clusters (Tan et al. 2005). One of the most popular internal indices is the *sum of squared errors* (SSE) — a cohesion measure based on a distance metric, calculated as:

$$SSE = \sum_{i=1}^{k} \frac{1}{2N_i} \sum_{o_x \in C_i} \sum_{o_y \in C_i} dist(o_x, o_y)^2,$$

where k is the number of clusters, N_i is the number of objects in cluster C_i, o_x and o_y are two objects assigned to cluster C_i, and $dist()$ is a distance function. The most popular separation measure is the *between group sum of squares* (SSB), calculated as the sum of squared distances of cluster centroids c_i to the overall mean c of all the objects:

$$SSB = \sum_{i=1}^{K} N_i dist(c_i, c)^2.$$

Cohesion and separation are often combined to ensure high intra-cluster and low inter-cluster similarity. An example of such a combination is the *silhouette* coefficient (Rousseeuw 1987). For an object o, the silhouette coefficient is computed as:

$$Silhouette(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}},$$

where $a(o)$ is the average dissimilarity of o with all other data within the same cluster and $b(o)$ is the minimal of average distances between o and any cluster that does not contain o. The value of the silhouette coefficient varies between -1 and 1, with 1 being the most desirable value.

The internal measures discussed so far were designed for partitional clustering algorithms, such as k-means. For hierarchical clustering, one of the most popular internal indices is the *cophenetic correlation* (Sokal & Rohlf 1962). Cophenetic correlation allows to determine which type of hierarchical clustering technique (single link, complete link, average link, etc.) best fits a given set of objects.

If a clustering algorithm can be tested on a benchmark dataset that contains objects with class labels, external measures can be used to validate the clustering. *External measures* (also called *external indices* or *supervised measures*) verify how well a clustering structure matches an external structure. To compare these two partitionings, supervised measures utilize user-supplied class labels which provide the correct clustering of objects in a benchmark dataset.

One of the simplest external indices is the *purity* measure (Zhao & Karypis 2002), which evaluates the degree to which a cluster contains documents from a single category. For a given cluster C_i of size N_i and a set of classes L, the purity of C_i is calculated as:

$$Purity(C_i, L) = \frac{1}{N_i} \max_{l \in L}(N_i^l)$$

where N_i^l represents the number of documents from cluster C_i assigned to category l and $\max_l(N_i^l)$ is the number of objects from the dominant category in cluster C_i. For k clusters containing N

objects in total, the overall purity of a clustering is defined as:

$$Purity(C, k) = \sum_{i=1}^{k} \frac{N_i}{N} Purity(C_i).$$

A measure with similar properties which can be used to externally evaluate clusters is *entropy* (Tan et al. 2005). Although purity and entropy are simple and transparent methods, it is easy to achieve high purity or entropy when the number of clusters is large; both measures achieve highest possible values if each document is assigned to its own single-element cluster. To trade off the quality of a clustering against the number of clusters, a measure called *normalized mutual information* (NMI) can be used. NMI is calculated as:

$$NMI(C, K) = \frac{I(C, K)}{[H(C) + H[K]]/2},$$

where $I()$ is the *mutual information* measure (Cover & Thomas 1991) and $H()$ is the entropy. Because in NMI mutual information is normalized, it can be used to compare clusterings with different numbers of clusters.

An alternative approach to evaluating clusterings using external information involves interpreting a clustering as a series of decisions concerning the assignment of an object to a cluster. From this point of view, a true positive (TP) decision is one that assigns two similar objects to the same cluster, whereas a true negative (TN) decision assigns two dissimilar objects to different clusters. Analogously, false positive (FP) and false negative (FN) decisions are those that assign two similar objects to different clusters and two dissimilar objects to the same cluster, respectively. With such an interpretation of different clustering decisions, the *rand index* (RI) (Rand 1971) was proposed as:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}.$$

A similar measure based on the Jaccard coefficient (Tan et al. 2005) can be computed as:

$$Jaccard = \frac{TP}{TP + FP + FN}.$$

In many scenarios it is important to differentiate the cost of making false positive and false negative decisions. The most popular evaluation measure that allows to do this is the *F-score* (Baeza-Yates & Ribeiro-Neto 1999), which penalizes false negatives according to a user-specified factor $\beta > 0$:

$$F_\beta = \frac{(\beta^2 + 1) Precision Recall}{\beta^2 Precision + Recall},$$

where

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN}.$$

External indices provide a transparent comparison between an algorithm's output and the desired clustering. Moreover, supervised measures allow to differentiate the importance of false positive and false negative errors. Unfortunately, in many applications external benchmarks are unavailable or do not fully encapsulate the problems of real-world scenarios. For this reason, clustering algorithms designed to tackle large datasets will most probably require unsupervised validation measures. In the field of XML clustering by structure the most popular evaluation methods are SSE/SSB (Flesca et al. 2005, Hwang & Ryu 2010, Nayak & Iryadi 2006, Aggarwal et al. 2007), precision and recall (Candillier et al. 2006, Dalamagas et al. 2006, Aggarwal et al. 2007, Brzezinski et al. 2011, Piernik et al. 2014), the F-score (Nayak & Iryadi 2006, Tran et al. 2008), and variations of the purity measure (Lee et al. 2002, Nierman & Jagadish 2002, Leung et al. 2005, Candillier et al. 2006, Helmer 2007).

8 Approaches

The available document representations, similarity measures, and clustering methods constitute a myriad of possibilities to create structural XML clustering algorithms. Throughout the last 10 years, several solutions utilizing the discussed representations and methods have been proposed. In this section, we review existing structural clustering algorithms and summarize their properties. The discussed algorithms will be analyzed in four groups: *tree-edit distance*, *substructural similarity*, *level similarity*, and *other* approaches.

8.1 Tree-edit distance approaches

Out of several tree-edit distance algorithms (Selkow 1977, Zhang & Shasha 1989, Chawathe et al. 1996, Chawathe 1999) Nierman and Jagadish (Nierman & Jagadish 2002) put forward an algorithm designed specifically for XML documents. In their approach, the authors propose to represent XML documents as a labeled ordered trees where inner nodes are tags and leaf nodes are tags or attributes. The algorithm allows three basic edit operations: relabeling, leaf insertion, and leaf deletion. In contrast to many other tree-edit distance methods, the authors propose two additional operations: subtree insertion and subtree deletion. Subtree insertions and deletions are limited only to subtrees that are already contained in the source/destination tree, i.e., if all nodes of the inserted/deleted subtree occur in the source/destination tree with the same parent-child relationships and the same sibling order (Nierman & Jagadish 2002). Using the proposed tree-edit distance measure, the authors calculate dissimilarities between XML documents and cluster them with a hierarchical agglomerative algorithm.

An approach that also utilizes the edit distance, called *binary branch distance*, was put forward by Yang et al. (Yang et al. 2005). Although originally proposed for XML query similarity search, the binary branch distance is a dissimilarity measure which is also suited for clustering algorithms. In this method, Yang et al. propose to transform each XML document into a normalized binary tree. Each node from the original document tree has exactly 2 children in the binary tree: the left child in the binary tree corresponds to a child in the original tree while the right child corresponds to the right sibling in the original tree. If a node has no child or sibling, a special node ϵ is added. Next, the number of occurrences of each unique triplet (node, left child, right child) is encoded in a vector called the *binary branch vector*. By subtracting corresponding positions of two binary branch vectors one can calculate the dissimilarity of two documents, called the binary branch distance. Yang et al. stated and proved that the binary branch distance forms a lower bound for tree-edit distance. In contrast to tree-edit distance, the binary branch distance can be calculated in linear time and is one of the most efficient ways of comparing full tree structures.

Dalamagas et al. (Dalamagas et al. 2006) proposed to summarize XML tree structures by eliminating the nesting of identical subtrees and the repetition of full paths. The resulting document representation, similar to the dataguide structure (Goldman & Widom 1997), considerably simplifies document processing. The authors propose to compare summarized documents using the edit distance algorithm proposed by Chawate (Chawathe 1999) and cluster them with a single link hierarchical method. Moreover, Dalamagas et al. present a way to adopt the C-index (Hubert & Levin 1976) in a hierarchical clustering procedure to estimate the optimal number of clusters.

More recently, Tekli and Chbeirb (Tekli & Chbeir 2012) put forward an XML clustering algorithm which employs a technique similar to that proposed by Nierman and Jagadish. The authors propose a tree-edit distance measure that allows for subtree deletions and insertions. To perform a subtree operation, the algorithm requires the presence of the analyzed subtree in both documents. However, in contrast to the approach proposed by Nierman and Jagadish, the subtrees found in both documents do not need to be identical. This way, when a subtree insertion or deletion is performed, its cost is proportional to the similarity of the two analyzed subtrees, which is calculated as a weighted average of their *structural commonality* and *semantic* *resemblance*. Structural commonality takes into account nodes with the same label, depth, and relative order in both subtrees, while semantic resemblance makes use of the WordNet thesaurus and the cosine measure. By calculating subtree similarity, in contrast to finding contained-in relations like Nierman and Jagadish, the described approach is able to detect a wider set or similarities and, thus, possibly produce more accurate clusters. The proposed tree-edit distance was used with a single link hierarchical clustering method.

8.2 Substructural similarity approaches

A compromise between fast tag-based methods and accurate edit distance approaches is provided by algorithms that use XML paths for clustering. For example, the *PBClustering* algorithm (Leung et al. 2005) describes each XML document as a set of XPath queries from root to leaf. Since there can be a large number of such queries, the authors propose to use only the maximal frequent ones, called *Common XPaths* (CXP). A frequent path is defined as a path that occurs *minSup* or more times in a dataset, where *minSup* is a user defined minimum support level. A maximal path is defined as a path which, in a given dataset, does not have any superpaths. After extracting all CXPs, each document is encoded as a bit vector. Each bit in this vector corresponds to a CXP and is set to 1 if a document contains that CXP or 0 otherwise. Finally, a similarity matrix is created by comparing each pair of bit vectors and clustering is performed with an agglomerative algorithm.

Another path-based XML clustering algorithm was proposed by Rafiei et al. (Rafiei et al. 2006). In this algorithm, for each document all paths beginning at the root node are extracted. A set consisting of these paths constitutes a representation for each document. Next, pairwise similarity between all documents is computed. The authors suggest to use one of the existing similarity measures dedicated for sets, like the Jaccard coefficient. Finally, after computing the similarity between all documents in the dataset, documents are clustered using the AHC algorithm.

Costa et al. (Costa et al. 2004) proposed an algorithm, called *XRep*, which also calculates similarity between trees with Jaccard coefficient using sets of tags or paths. In this approach, every cluster has a representative which summarizes this cluster. This representative is a tree which has the lowest distance from all trees in the cluster. Its lower-bound is an intersection of all documents in the cluster and its upper-bound is their union. Clustering is performed with the AHC algorithm with inter-cluster similarity calculated based on the cluster representatives.

Vercoustre et al. (Vercoustre et al. 2006) proposed a family of path-based representations suited for both, structure and structure and content analysis. In each of these representations, paths are encoded into vectors of term frequencies. When defining paths, the authors consider four options: limiting their length, using different start and end nodes, including their textual content and attribute nodes, or finally, using only textual nodes. Each representation also holds the number of occurrences of each term. After transforming a dataset into a chosen representation, the authors reduce the number of paths and resolve any possible dependencies between them. The reduction is performed on two levels: textual — stemming, stop lists, removing words shorter than four; and structural — tag generalization. Path dependencies, i.e., superpaths and subpaths, are resolved by decomposing the superpaths into smaller paths. Clustering is performed with the k-means algorithm, where each document is represented as a term frequency vector scaled with the inverse document frequency (TF-IDF). Centroids are computed as a sum of all vectors in a given cluster and similarity is evaluated using the Euclidean distance.

Two similar approaches were proposed in Tran et al. (2007) and Kutty et al. (2007). The authors utilize a two-phase approach with each document represented as a set of full paths (Tran et al. (2007)) or a set of subtrees (Kutty et al. (2007)). In the first phase, documents are clustered incrementally into k' clusters, where $k' > k$. Each document is compared with every existing cluster and is assigned to the most similar cluster, given that the similarity exceeds a user-defined threshold μ. If not, then the document forms a separate cluster and becomes its representative. In the second phase, the obtained k' clusters are further merged according to their similarity, until they form k clusters. The similarity between the documents and the cluster representatives is handled differently in these approaches. The authors in Kutty et al. (2007) use the Jaccard coefficient, while the authors in Tran et al. (2007) propose an original measure which calculates the joint similarity of all paths in both documents.

A different approach, called S-GRACE, was proposed by Lian et al. (Lian et al. 2004). In this solution, documents are summarized into graph structures, called *s-graphs*, which contain all edges but only distinct nodes from the original document tree. Next, all s-graphs are encoded as bit vectors, where each position in a vector reflects the presence or absence of a corresponding edge in a given s-graph. Since one s-graph may correspond to many documents, each vector is additionally associated with a list of documents corresponding to a given s-graph. Such a structure representing the whole dataset is called *SG*. After constructing the SG, all pairs of s-graphs are compared using a distance measure based on the percentage of common edges between them. The result of such a comparison (e.g., a distance matrix) may be used by any applicable clustering algorithm. However, the authors propose to use the ROCK algorithm (Guha et al. 2000), due to the fact that binary values in SG should be considered as categorical rather than numerical. The algorithm starts with each s-graph representing a separate cluster. In each consecutive step, a pair of closest clusters is merged. The closeness of two clusters is calculated based on the number of common neighbors of all s-graphs from these clusters; the neighborhood is defined by a user-specified maximal distance threshold between two s-graphs. Based on these notions, s-graphs are grouped until reaching the desired number of clusters.

An approach similar to S-GRACE was presented by Aïtelhadj et al. (Aïtelhadj et al. 2012). The authors propose to transform XML documents into tree summaries by merging all repeating elements at each level of a document into a single node. After the initial transformation, the summaries are further processed as sets of full paths and are clustered with an iterative algorithm. The first document forms the first cluster. Afterwards, each new document is either assigned to one of the existing clusters or constitutes a new one if it does not fit to any of the existing clusters. The document fits into a cluster if it achieves a given threshold of similarity with that cluster's centroid, i.e., the most representative document chosen as the one with the highest similarity with all other documents in this cluster. The similarity between two document trees is computed as a weighted sum of similarity between all of their paths starting at the root node. Such a clustering procedure ensures that new documents do not require recalculating the centroids of all clusters.

Another interesting approach was introduced in the *XProj* framework (Aggarwal et al. 2007). The authors propose a clustering algorithm that uses frequent substructures (tree edges) as patterns. Initially, the document set is randomly divided into k partitions of equal size. Next, sets of frequent edges (cluster representatives) are mined from these partitions and later used for defining similarity among documents. A distance between a document and a set of representatives is defined as the fraction of edges in the document which also occur in any of the representatives. According to the computed distances, documents are reassigned to clusters with the most similar representatives. In subsequent iterations, the algorithm mines for new frequent edges and repeats the clustering process until it converges or reaches the maximum number of iterations.

Another method which utilizes frequent substructures was proposed by Brzezinski et al. (Brzezinski et al. 2011). In their algorithm, the authors propose a three-step pattern-based approach. First, the algorithm mines the whole dataset for maximal frequent subtrees, which serve as patterns. Later, the patterns are grouped into k clusters using the AHC algorithm, with the number of common documents as a similarity measure. Finally, each document is assigned to its proper cluster based on the number of common patterns. This approach was tested and further generalized into a generic pattern-based framework called *XPattern* (Piernik et al. 2014). The framework formalizes four main steps of a pattern-based clustering methodology: document transformation, pattern mining, pattern clustering and document assignment. This approach does not imply any particular representation or pattern definition. However, the authors tested several different pattern representations and the results of the study suggest that frequent paths provide a good balance between information saturation and efficiency.

8.3 Level similarity approaches

Tag-based methods are considered to be the simplest and least accurate algorithms for clustering XML documents by structure. However, Nayak proposed a new tag-based approach, called *XCLS*, in which the author included additional information about the level of each tag in the document tree (Nayak 2008). Therefore, the method incorporated more structural information than regular tag-based approaches, yet preserved their simplicity. In this method, each document is represented as a *LevelStructure* — a vector whose cells correspond to consecutive levels in the document tree. In each cell, there is a vector containing distinct tags which appear in the document at the corresponding level. The author also proposed a weighted similarity measure which measures the co-occurrences of elements at corresponding levels, where a certain weight is associated with each level. With the given similarity measure, clusters are formed in an iterative manner. Each new XML document is placed in a cluster which contains the most similar documents.

The XCLS algorithm was enhanced by Alishahi et al. (Alishahi et al. 2010) who identified two main problems of the algorithm. The first problem concerns comparing two trees when the root node label of one tree does not appear anywhere in the other tree and vice versa. This always results in a similarity equal zero even if all other nodes match. The second problem is the lack of information about parent-child connections. Both problems were addressed in an improved version of the XCLS method called XCLS+ (Alishahi et al. 2010).

An approach complementary to XCLS and XCLS+ was proposed by Antonellis et al. in an algorithm called *XEdge* (Antonellis et al. 2008). In XEdge, the authors propose to represent XML documents as *LevelEdges*, structures similar to the LevelStructure (Nayak 2008). In contrast to LevelStructures, LevelEdges contain information about edges on each level of an XML document. This way, in addition to containing information about nodes, a LevelEdge encapsulates information about parent-child relationships between them. Such an approach can help to distinguish groups of similar documents not only in heterogeneous, but also homogeneous datasets. To further adapt XEdge to process both homogeneous and heterogeneous datasets, the authors propose two separate similarity measures. To compare documents from homogeneous sources, XEdge uses a measure which calculates a weighted sum of common edges on each level divided by a weighted sum of all unique edges on each level. For the differentiation of heterogeneous documents, the authors propose a measure analogous to that proposed in (Nayak 2008), which tries to match similar levels of two compared documents and calculate a weighted sum of common edges. To cluster a dataset of XML documents, XEdge uses a modified version of k-means which utilizes LevelEdge similarity to calculate the distance between two documents. As cluster representatives, the authors propose to use yet another leveled structure, in which each level contains all distinct edges in the corresponding level of all document representations in a given cluster.

In an attempt to minimize the processing time and memory requirements, Hadzic et al. (Hadzic et al. 2011) proposed to transform XML documents into a structure called *document structure model* (DSM). A DSM is a string vector that allows to encode the structure of documents from a given dataset in a concise and easily comparable manner. To achieve this, each document in a dataset is first pre-order traversed and string encoded using an algorithm proposed by Zaki (Zaki 2002). Next, the obtained flat document representations are "aligned" so that a position in the DSM corresponds to exactly the same level and position in each XML document in the dataset. The authors positively evaluated the proposed method against a tree-edit distance approach and have shown that almost any similarity measure and clustering approach can be used with the DSM representation.

8.4 Other approaches

A different vector-based approach for clustering XML documents was proposed by Candillier et al. (Candillier et al. 2006). The authors suggest to summarize each document with a set of attribute-value pairs. The attributes correspond to distinct features extracted from the documents: parent-child relations, next-sibling relations, paths, tags, and absolute node positions. The value of each attribute is the number of occurrences of the corresponding feature in a given document. After calculating feature vectors for all documents, clustering is performed with a variation of the expectation-maximization algorithm (Moon 1996). Since the number of attributes for a dataset can grow fairly large, the authors propose to carry out feature selection. To perform this, for each cluster, attributes are weighted by the ratio between local and global standard deviations of attribute values. Afterward, only a user-defined number of highest weighted attributes is kept for further processing.

Hagenbuchner et al. put forward an algorithm called *SOM-SD* (Hagenbuchner et al. 2006), which extends Kohonen's *self organizing map* (SOM) (Kohonen 1989) and clusters XML documents in an unsupervised fashion. For each document in a dataset, SOM-SD processes document tree nodes one at a time and maps them on the SOM neuron grid. A node is represented by a vector containing the node's label and the neuron grid coordinates of that node's direct offspring. For this reason, SOM-SD processes XML documents from leaf nodes to root nodes. After introducing all the documents to the SOM, similar documents are displayed at the same or close coordinates on the neuron grid. Apart from putting forward a clustering algorithm, the authors also propose evaluation measures suitable for documents mapped on the SOM grid. Furthermore, the same paper introduces a more general algorithm called *CSOM-SD*, which can be used to cluster XML documents with links or represented as graphs, e.g., s-graphs (Lian et al. 2004).

One of the most original approaches to clustering XML by structure was proposed by Flesca et al. (Flesca et al. 2005). In their solution, the authors omit not only the values of elements and attributes, but also their labels, leaving only the information about the order and depth of the nodes. Hence, this approach transforms each document into a time series, where a signal in a given moment in time corresponds to the depth of a node appearing at a particular position in the document tree. After the transformation, all documents are compared using discrete Fourier transform. The product of this comparison may be utilized by any applicable clustering algorithm.

Wang et al. put forward a method dedicated for very large XML datasets (Wang et al. 2006). The authors propose to transform each document into a set of paths and later encode each path as a single number. A set of such numbers becomes a document representation. A distance measure based on this representation calculates an absolute difference between all values of the compared documents. The documents are clustered with the k-means algorithm, according to the defined distance measure. In order to reduce the probability of the solution falling into a local optimum, the authors propose a modification of the k-means algorithm. After achieving convergence, random objects are relocated between the clusters and centroids are recalculated.

One of the more efficient XML clustering methods is the entropy-based clustering algorithm (Helmer 2007, Helmer et al. 2012). The main idea behind this approach is to compress structural information about documents, compare the lengths of the compressed files, and calculate the normalized compression distance (NCD) between each pair of documents. The calculated distances can be used by any similarity-based clustering algorithm. Moreover, this algorithm can work with any type of document representation. The definition of NCD allows such flexibility, as it is based on an approximation of the Kolmogorov complexity, which can be defined for any data object (Bennett et al. 1998). Although the algorithm can use any type of document representation, clustering results will differ depending on the selected representation. For this reason, the author analyzes four methods of extracting structural information - tags, pairs of tags, paths, and whole document trees. Compared with simple tag and edit distance algorithms,

Table 4 Summary of presented approaches

Approach	**Representation**	**Similarity measure**	**Clustering**
TED (Nierman & Jagadish 2002)	Rooted ordered labeled tree	Tree-edit distance	Hierarchical (AHC)
Binary Branch (Yang et al. 2005)	Binary branch vector	Binary branch distance	Unspecified
Structural Summaries (Dalamagas et al. 2006)	Graph	Tree-edit distance	Hierarchical (Single-link)
Subtree Commonalities and Label Semantics (Tekli & Chbeir 2012)	Rooted ordered labeled tree	Tree-edit distance	Hierarchical (Single-link)
PBClustering (Leung et al. 2005)	Set of XPath queries	Number of common XPath queries	Hierarchical (AHC)
Syntactic Similarities (Rafiei et al. 2006)	Set of root paths	Unspecified; any set-based measure (e.g., Jaccard Coefficient)	Hierarchical (AHC)
XRep (Costa et al. 2004)	Set of tags/paths	Jaccard coefficient	Hierarchical (AHC)
Path Family (Vercoustre et al. 2006)	Path frequency vector	Euclidean distance	K-means
Path Similarity (Tran et al. 2007, Kutty et al. 2007)	Set of full paths	Based on Jaccard coefficient	Iterative
S-GRACE (Lian et al. 2004)	S-graph	Percentage of common edges	ROCK
Structural Similarity (Aïtelhadj et al. 2012)	Tree summary	Weighted path similarity	Iterative
XProj (Aggarwal et al. 2007)	Set of maximal frequent edges	Common maximal frequent edges	Partition-based
XPattern (Brzezinski et al. 2011)	Frequent features	Common frequent features	Hierarchical (AHC)
XCLS/XCLS+ (Nayak 2008, Alishahi et al. 2010)	Vector of tags with tree levels	Co-occurrence of tags at corresponding levels	Hierarchical
XEdge (Antonellis et al. 2008)	Vector of edges with tree levels	Co-occurrence of edges at corresponding levels	K-means
DSM (Hadzic et al. 2011)	String encoded trees	Jaccard or correlation coefficient	Unspecified
SSC (Candillier et al. 2006)	Feature vector	Cluster membership probability	EM
SOM-SD (Hagenbuchner et al. 2006)	Set of node vectors	Neuron grid coordinates	SOM
DFT (Flesca et al. 2005)	Time series	Discrete Fourier transform	Unspecified
ICX (Wang et al. 2006)	Set of numerically encoded paths	Absolute difference between all values in the sets	K-means
Entropy-based clustering (Helmer 2007, Helmer et al. 2012)	Compressed tree	Normalized compression distance	Hierarchical (AHC)
Weighted Common Structure (Hwang & Ryu 2010, 2004)	Set of frequent paths	Cluster allocation profit	CLOPE

the entropy-based approach requires less time and achieves similar or better clustering accuracy, depending on the selected representation and compression algorithm.

Hwang and Ryu (Hwang & Ryu 2004, 2010) proposed to model XML documents as transactions, where transaction items are frequent paths found in the documents. With such a representation, the authors use a modified version of the CLOPE algorithm (Yang et al. 2002). To assign documents to clusters, CLOPE uses the rate of common paths to determine cluster allocation profit. To confirm that the cluster allocation profit ensures high cluster cohesion, the authors propose to weight frequent structures according to their support and a user-specified value. It is worth noting that this approach does not require pairwise document comparisons and, therefore, can be used for large, schema-less datasets.

9 Discussion

In the previous section, we presented 23 state-of-the-art approaches to clustering XML documents by structure. Some of these methods derive from existing branches of data mining, while others present new concepts, specific to semistructured data. In order to facilitate the comparison of the presented methods, we have gathered them in Table 4, where each approach is divided into three main steps according to the clustering methodology described in Section 3.

Studying the approaches presented in Table 4, we can analyze the applicability of the algorithms in different real-world scenarios. When dealing with homogeneous datasets, complex representations and similarity measures need to be applied in order to differentiate between groups of documents. In such cases, tree representations and edit-distance-based measures are advised. However, when facing a large collection of documents, clustering using tree-based approaches may be infeasible. When such is the case, one should consider lighter representations, like paths, and combine them with simple, frequency-based similarity measures. Finally, when groups of documents in the collection are easily separable, e.g., originate from heterogeneous data sources, simple tag- or even metadata-based approaches may be sufficient.

All of the analyzed approaches were classified according to the methods they utilize. In Figure 9 we present a taxonomy illustrating this classification with respect to different representations and similarity measures. The names of the approaches in the diagram correspond with the names used in Table 4. Each approach in this diagram is depicted by an ellipsis and the shades reflect the categorization proposed in Section 8: white — substructural similarity, light gray — tree-edit distance, dark gray — level similarity, black — other.

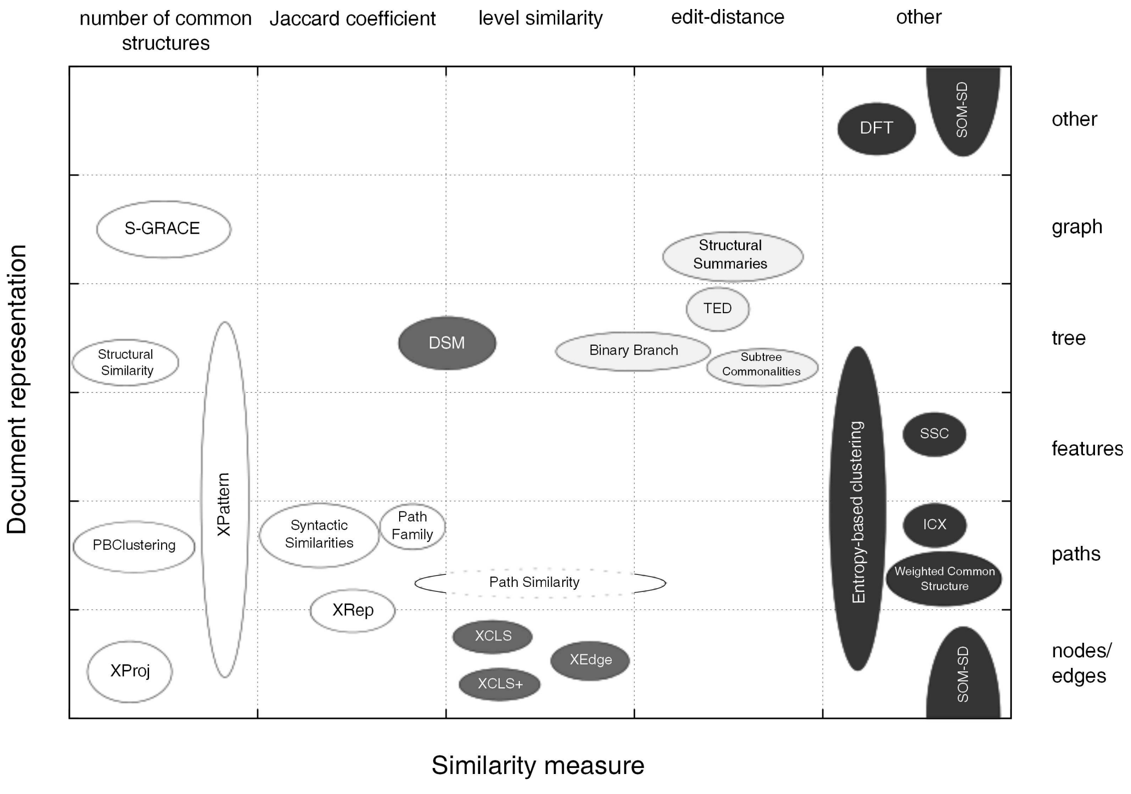


Figure 9 Taxonomy of XML structural clustering approaches

The diagram indicates that the most popular and studied groups of approaches are the ones using edit distance for trees and Jaccard coefficient for paths. Such prominent use of tree-edit distance methods can be explained by the fact that trees are the most natural representation for XML documents and have been analyzed long before the introduction of XML. On the other hand, paths are much easier to process while still retaining high structural information level. Interestingly, inasmuch as the presented representations are mainly adoptions of well-established solutions, the similarity measures are often original methods, which is evidenced by the difference in the number of unclassifiable approaches (category *other*).

It is also worth noticing that even though paths are extensively used across the presented representations, most of the approaches do not utilize their full potential when calculating similarity. The vast majority of algorithms treats paths as terms and evaluate similarity based just on their presence, neglecting the actual structural similarity between them. This issue is evidenced by the blank cross-section between path representation and edit distance similarity on the diagram. Utilizing additional structural information when comparing paths could significantly improve the quality of clustering while retaining reasonable complexity.

Another apparent issue stemming directly from the diagram is the fact that there are very few approaches using feature representation. XPattern and Entropy-based clustering both allow it, but are more general solutions. In fact, the only approach dedicated for this type of representation is SSC. Because of the extreme information reduction, the feature representation usually produces results of low quality, however, thanks to very low complexity, it may be one of the only feasible options when dealing with big data.

As one can see, the diagram in Figure 9 excludes clustering algorithms and focuses only on representations and similarity measures. This is because all of the clustering algorithms used in the described approaches are general, not XML-specific. Furthermore, reasons for selecting a specific clustering algorithm are rarely reported. It would be very interesting to see how different algorithms influence the whole XML clustering process and whether certain clustering algorithms suit some representations or similarity measures better than others. However, such an analysis is still to be performed and constitutes an opportunity for novel research. A possible explanation to the highlighted lack of interest in clustering algorithms could lie in the fact that nearly all of the presented approaches, either explicitly or implicitly, follow the same general framework presented in Section 3, which diminishes the role of the clustering algorithm. It would be interesting to see more clustering algorithms designed specifically for XML.

When developing a solution in the XML clustering domain, researchers face another problem completely unrelated to the scientific task — shortage of publicly available, real-world datasets. Such a situation makes it very difficult to analyze newly proposed methods, not to mention compare them with existing approaches. The comparison is even more difficult to perform since usually not only the datasets are unavailable but also the implementations of the competitive algorithms. This situation partially stems from the fact that, even though XML clustering has many applications as described in Section 2, the actual application rate of the proposed methods is very low. Clearly, there is a strong need for some kind of XML clustering platform or a repository where researchers could store and test their algorithms and analysts could post real datasets and process them with different methods. This would greatly facilitate the comparison of existing methods and allow easy access to the newest solutions to real XML-related problems. This could be achieved either by a separate tool/platform or a set of extensions to existing data mining tools, e.g., R[2], Weka[3], or RapidMiner[4]. Currently, neither publicly available research tools nor commercial systems dedicated for XML clustering by structure exist.

[2]http://www.r-project.org/
[3]http://www.cs.waikato.ac.nz/ml/weka/
[4]http://rapidminer.com/

10 Conclusions

In this paper, we have conducted a survey on structure-based XML document clustering. First, we decomposed the problem into three core sub-problems: document representation, document similarity, and clustering algorithm, and separately analyzed each of these tasks. Next, we analyzed 23 state-of-the-art approaches highlighting the techniques used at every step of the clustering process. Each of the described approaches can be considered either a milestone in the domain, a unique and interesting solution, or a significant improvement over an existing method. By conducting an extensive, up-to-date review we hope to provide a valuable resource for researchers and practitioners in fields related to XML processing.

The performed analysis highlights open issues in the field of XML clustering by structure. Firstly, a strong dependency between the used similarity measures and document representations can be noticed. Similarity between documents represented with paths is usually evaluated with simple occurrence counting, while tree structures are compared using tree-edit distance measures. Consequently, there is a room for developing more complex path measures which would take into account not only their presence but also their structural relationships. Furthermore, the last step of the clustering process leaves room for further developments, as currently only general, non-XML-specific clustering algorithms are in use. Finally, the most important problem in the XML clustering domain lies in the shortage of publicly available real-world datasets and the absence of a common development and testing platform. We believe that addressing that last problem would be of high practical value to the research community and could improve the dissemination of newly developed approaches.

Acknowledgement

The authors wish to thank the editor and the anonymous reviewers for their useful comments and suggestions. This work was partly supported by the Polish National Science Center under Grant No. DEC-2011/01/B/ST6/05169.

References

Achard, F., Vaysseix, G. & Barillot, E. (2001), ‘XML, bioinformatics and data integration’, *Bioinformatics* **17**(1), 115–125.

Aggarwal, C. C., Ta, N., Wang, J., Feng, J. & Zaki, M. (2007), XProj: a framework for projected structural clustering of XML documents, *in* ‘Proc. 13th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.’, KDD’07, pp. 46–55.

Aggarwal, C. C. & Zhai, C., eds (2012), *Mining Text Data*, Springer.

Aïtelhadj, A., Boughanem, M., Mezghiche, M. & Souam, F. (2012), ‘Using structural similarity for clustering XML documents’, *Knowl. Inf. Syst.* **32**(1), 109–139.

Algergawy, A., Mesiti, M., Nayak, R. & Saake, G. (2011), ‘XML data clustering: An overview’, *ACM Comput. Surv.* **43**(4), 25:1–25:41.

Alishahi, M., Naghibzadeh, M. & Aski, B. S. (2010), ‘Tag name structure-based clustering of XML documents’, *Int. J. Comput. Elect. Eng.* **2**(1), 119–126.

Andreopoulos, B., An, A., Wang, X. & Schroeder, M. (2009), ‘A roadmap of clustering algorithms: finding a match for a biomedical application.’, *Brief. Bioinform.* **10**(3), 297–314.

Ankerst, M., Breunig, M. M., Kriegel, H.-P. & Sander, J. (1999), ‘OPTICS: ordering points to identify the clustering structure’, *SIGMOD Rec.* **28**(2), 49–60.

Antonellis, P., Makris, C. & Tsirakis, N. (2008), XEdge: clustering homogeneous and heterogeneous XML documents using edge summaries, *in* 'Proc. 2008 ACM Symp. App. Comp.', SAC'08, pp. 1081–1088.

Baeza-Yates, R. A. & Ribeiro-Neto, B. A. (1999), *Modern Information Retrieval*, ACM Press / Addison-Wesley.

Bennett, C. H., Gacs, P., Li, M., Vitanyi, P. M. B. & Zurek, W. H. (1998), 'Information distance.', *IEEE Trans. Inf. Theory* **44**(4), 1407–1423.

Bertino, E., Guerrini, G. & Mesiti, M. (2008), 'Measuring the structural similarity among XML documents and DTDs', *J. Intell. Inf. Syst.* **30**(1), 55–92.

Brzezinski, D., Lesniewska, A., Morzy, T. & Piernik, M. (2011), 'XCleaner: A new method for clustering XML documents by structure', *Control and Cybernetics* **40**(3), 877–891.

Buttler, D. (2004), A short survey of document structure similarity algorithms, *in* 'Proc. 5th Int. Conf. Internet Comp.', pp. 3–9.

Candillier, L., Tellier, I. & Torre, F. (2006), Transforming XML trees for efficient classification and clustering, *in* 'Proc. 4th Int. Conf. of the Initiative for the Evaluation of XML Retrieval', INEX'05, pp. 469–480.

Chawathe, S. S. (1999), Comparing Hierarchical Data in External Memory, *in* 'Proc. 25th Int. Conf. on Very Large Data Bases', VLDB'99, pp. 90–101.

Chawathe, S. S., Rajaraman, A., Garcia-Molina, H. & Widom, J. (1996), Change detection in hierarchically structured information, *in* 'Proc. 1996 ACM SIGMOD Int. Conf. Manage. Data', SIGMOD '96, pp. 493–504.

Chemical Markup Language - CML (1995).
URL: *http://www.xml-cml.org/*

Costa, G., Manco, G., Ortale, R. & Tagarelli, A. (2004), A tree-based approach to clustering XML documents by structure, *in* 'Proc. 8th European Conf. on Principles and Practice of Knowl. Disc. in Databases', PKDD '04, pp. 137–148.

Cover, T. M. & Thomas, J. A. (1991), *Elements of information theory*, Wiley-Interscience.

Crescenzi, V., Mecca, G. & Merialdo, P. (2001), RoadRunner: Towards automatic data extraction from large web sites, *in* 'Proc. 27th Int. Conf. on Very Large Data Bases', VLDB'01, pp. 109–118.

Crescenzi, V., Merialdo, P. & Missier, P. (2005), 'Clustering web pages based on their structure', *Data Knowl. Eng.* **54**(3), 279–299.

Dalamagas, T., Cheng, T., Winkel, K.-J. & Sellis, T. (2006), 'A methodology for clustering XML documents by structure', *Inf. Syst.* **31**(3), 187–228.

Dalamagas, T., Cheng, T., Winkel, K.-J. & Sellis, T. K. (2004), Clustering XML documents by structure, *in* 'Proc. 3rd Helenic Conference on AI', SETN'04, pp. 112–121.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977), 'Maximum Likelihood from Incomplete Data via the EM Algorithm', *J. Roy. Stat. Soc. B Met.* **39**(1), 1–38.

Dolin, R. H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F. M., Biron, P. V. & Shvo, A. S. (2006), 'HL7 clinical document architecture, release 2', *J. Am. Med. Inform. Assoc.* **13**(1), 30–39.

Doucet, A. & Lehtonen, M. (2006), Unsupervised classification of text-centric XML document collections., *in* 'Proc. 5th Int. Conf. of the Initiative for the Evaluation of XML Retrieval', INEX'06, pp. 497–509.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, *in* 'Proc. 2nd Int. Conf. Knowl. Disc. Data Min.', KDD'96, pp. 226–231.

Flesca, S., Manco, G., Masciari, E., Pontieri, L. & Pugliese, A. (2002), Detecting structural similarities between XML documents, *in* 'Proc. 5th Int. Workshop on the Web and Databases', WebDB'02, pp. 55–60.

Flesca, S., Manco, G., Masciari, E., Pontieri, L. & Pugliese, A. (2005), 'Fast detection of XML structural similarity', *IEEE Trans. Knowl. Data Eng.* **17**(2), 160–175.

Goldman, R. & Widom, J. (1997), DataGuides: Enabling query formulation and optimization in semistructured databases, Technical Report 1997-50, Stanford InfoLab.

Guerrini, G., Mesiti, M. & Sanz, I. (2007), *Web Data Management Practices: Emerging Techniques and Technologies*, Idea Group Inc (IGI), chapter 3. An Overview of Similarity Measures for Clustering XML Documents.

Guha, S., Rastogi, R. & Shim, K. (1998), 'CURE: an efficient clustering algorithm for large databases', *SIGMOD Rec.* **27**(2), 73–84.

Guha, S., Rastogi, R. & Shim, K. (2000), 'ROCK: A robust clustering algorithm for categorical attributes', *Inf. Syst.* **25**(5), 345–366.

Hadzic, F., Hecker, M. & Tagarelli, A. (2011), XML document clustering using structure-preserving flat representation of XML content and structure, *in* 'Proc. 7th Int. Conf. Advanced Data Mining and Applications', ADMA'11, pp. 403–416.

Hagenbuchner, M., Sperduti, A., Tsoi, A. C., Trentini, F., Scarselli, F. & Gori, M. (2006), Clustering XML documents using self-organizing maps for structures, *in* 'Proc. 4th Int. Conf. on Initiative for the Evaluation of XML Retrieval', INEX'05, pp. 481–496.

Han, J. (2005), *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers Inc.

Hartigan, J. A. & Wong, M. A. (1979), 'A K-means clustering algorithm', *Appl. Stat.-J. Roy. St. C* **28**, 100–108.

Helmer, S. (2007), Measuring the structural similarity of semistructured documents using entropy, *in* 'Proceedings of the 33rd International Conference on Very Large Data Bases', VLDB '07, VLDB Endowment, pp. 1022–1032.

Helmer, S., Augsten, N. & Böhlen, M. (2012), 'Measuring structural similarity of semistructured data based on information-theoretic approaches', *VLDB J.* **21**(5), 677–702.

Hubert, L. J. & Levin, J. R. (1976), 'A general statistical framework for accessing categorical', *Psychol. Bull.* **83**, 1072–1082.

Husek, D., Pokorny, J., Rezankova, H. & Snasel, V. (2007), *Web Data Management Practices: Emerging Techniques and Technologies*, Idea Group Inc (IGI), chapter 1. Data Clustering: From Documents to the Web.

Hwang, J. H. & Ryu, K. H. (2004), A new XML clustering for structural retrieval, *in* 'Proc. 23rd Int. Conf. on Conceptual Modeling', ER'04, pp. 377–387.

Hwang, J. H. & Ryu, K. H. (2010), 'A weighted common structure based clustering technique for XML documents', *J. Syst. Software* **83**(7), 1267–1274.

Jain, A. K., Murty, M. N. & Flynn, P. J. (1999), 'Data clustering: A review', *ACM Comput. Surv.* **31**(3), 264–323.

Johnson, S. (1967), 'Hierarchical clustering schemes', *Psychometrika* **32**, 241–254.

Kohonen, T. (1989), *Self-organization and associative memory: 3rd edition*, Springer-Verlag New York, Inc.

Kutty, S., Nayak, R. & Li, Y. (2010), Utilising semantic tags in xml clustering, *in* 'Proc. 8th Int. Conf. on Initiative for the Evaluation of XML Retrieval', INEX'09, pp. 416–425.

Kutty, S., Nayak, R. & Li, Y. (2011), XML documents clustering using a tensor space model, *in* 'Proc. 15th Pacific-Asia Conf. on Advances in Knowl. Disc. and Data Mining', PAKDD'11, pp. 488–499.

Kutty, S., Tran, T., Nayak, R. & Li, Y. (2007), Clustering XML documents using closed frequent subtrees: A structural similarity approach, *in* 'Proc. 5th Int. Conf. of the Initiative for the Evaluation of XML Retrieval', INEX'06, pp. 183–194.

Kutty, S., Tran, T., Nayak, R. & Li, Y. (2008), Clustering xml documents using frequent subtrees., *in* 'Proc. 6th Int. Conf. of the Initiative for the Evaluation of XML Retrieval', INEX'07, pp. 436–445.

Lee, M. L., Yang, L. H., Hsu, W. & Yang, X. (2002), XClust: clustering XML schemas for effective integration, *in* 'Proc. 11th Int. Conf on Information and Knowledge Management', CIKM'02, pp. 292–299.

Lesniewska, A. (2009), Clustering XML documents by structure, *in* 'Proc. 13th East European Conference Advances in Databases and Information Systems', ADBIS'09, pp. 238–246.

Lesniewska, A. & Primke, K. (2008), Finding features on the basis of the structure of XML documents, Technical Report RA 16/08, Poznan University of Technology.

Leung, H.-P., Chung, K. F.-L., fai Chan, S. C. & Luk, R. W. P. (2005), XML document clustering using common XPath, *in* 'Prof. 2005 Int. Workshop on Challenges in Web Information Retrieval and Integration', WIRI'05, pp. 91–96.

Li, J., Liu, J., Liu, C., Wang, G., Yu, J. X. & Yangt, C. (2007), Computing structural similarity of source XML schemas against domain XML schema, *in* 'Proc. 19th Conf. Australasian Database', ADC'08, pp. 155–164.

Lian, W., Cheung, D. W.-l., Mamoulis, N. & Yiu, S.-M. (2004), 'An efficient and scalable algorithm for clustering XML documents by structure', *IEEE Trans. Knowl. Data Eng.* **16**(1), 82–96.

Mathematical Markup Language - MathML (1998).
URL: *http://www.w3.org/Math/*

Moon, T. K. (1996), 'The expectation-maximization algorithm', *IEEE Signal Processing Mag.* **13**(6), 47–60.

Nayak, R. (2008), 'Fast and effective clustering of XML data using structural information', *Knowl. Inf. Syst.* **14**(2), 197–215.

Nayak, R. & Iryadi, W. (2006), XMine: a methodology for mining XML structure, *in* 'Proc. 8th Asia-Pacific Web Conf. on Frontiers of WWW Research and Development', APWeb'06, pp. 786–792.

Nierman, A. & Jagadish, H. V. (2002), Evaluating structural similarity in XML documents, *in* 'Proc. 5th Int. Workshop on the Web and Databases', WebDB'02, pp. 61–66.

Pawlik, M. & Augsten, N. (2011), 'RTED: A robust algorithm for the tree edit distance', *Proc. VLDB Endow.* **5**(4), 334–345.

Piernik, M., Brzezinski, D. & Morzy, T. (2014), 'Clustering XML documents by patterns', *Knowl. Inf. Syst.* **review in progress**.

Pospech, T. (2009), *GML - Geography Markup Language*, GRIN Verlag.

Rafiei, D., Moise, D. L. & Sun, D. (2006), Finding syntactic similarities between XML documents, *in* 'Proc. 17th Int. Conf. on Database and Expert Systems Applications', DEXA '06, pp. 512–516.

Rand, W. M. (1971), 'Objective criteria for the evaluation of clustering methods', *J. Am. Stat. Assoc.* **66**(336), 846–850.

Rousseeuw, P. (1987), 'Silhouettes: a graphical aid to the interpretation and validation of cluster analysis', *J. Comput. Appl. Math.* **20**(1), 53–65.

Sankoff, D. & Kruskal, J. B. (2000), *Time warps, string edits, and macromolecules*, Cambridge University Press.

Selkow, S. (1977), 'The tree-to-tree editing problem', *Inform. Process. Lett.* **6**(6), 184–186.

Sokal, R. R. & Rohlf, F. J. (1962), 'The comparison of dendrograms by objective methods', *Taxon* **11**(2), 33–40.

Somervuo, P. & Kohonen, T. (2000), Clustering and visualization of large protein sequence databases by means of an extension on the self-organizing map, *in* 'Proc. 3rd Int. Conf. on Discovery Science', DS'00, pp. 76–85.

Tagarelli, A. & Greco, S. (2010), 'Semantic clustering of xml documents', *ACM Trans. Inf. Syst.* **28**(1), 3:1–3:56.

Tai, K.-C. (1979), 'The tree-to-tree correction problem', *J. ACM* **26**(3), 422–433.

Tan, P.-N., Steinbach, M. & Kumar, V. (2005), *Introduction to Data Mining*, Addison-Wesley.

Tekli, J. & Chbeir, R. (2012), 'A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics', *J. Web Sem.* **11**, 14–40.

Tekli, J., Chbeir, R. & Yetongnon, K. (2009), 'Survey: An overview on XML similarity: Background, current trends and future directions', *Comput. Sci. Rev.* **3**(3), 151–173.

Theobald, M. (2003), Exploiting structure, annotation, and ontological knowledge for automatic classification of XML data, *in* 'Proc. WebDB Workshop', WebDB'03, pp. 1–6.

Tran, T., Nayak, R. & Bruza, P. (2007), Document clustering using incremental and pairwise approaches., *in* 'Proc. 5th Int. Conf. of the Initiative for the Evaluation of XML Retrieval', Vol. 4862 of *INEX'06*, pp. 222–233.

Tran, T., Nayak, R. & Bruza, P. (2008), Combining structure and content similarities for XML document clustering., *in* 'Proc. 7th Australasian Data Mining Conf.', AusDM'08, pp. 219–226.

Vakali, A., Pallis, G. & Angelis, L. (2007), *Web Data Management Practices: Emerging Techniques and Technologies*, Idea Group Inc (IGI), chapter 2. Clustering Web Information Services.

Vercoustre, A.-M., Fegas, M., Gul, S. & Lechevallier, Y. (2006), A flexible structured-based representation for XML document mining, *in* 'Proc. 4th Int. Conf. of the Initiative for the Evaluation of XML Retrieval', INEX'05, pp. 443–457.

Viyanon, W., Madria, S. K. & Bhowmick, S. S. (2008), XML data integration based on content and structure similarity using keys, *in* 'Proc. OTM 2008 Confederated International Conferences CoopIS, DOA, GADA, IS, and ODBASE', OTM'08, pp. 484–493.

Wang, T., Liu, D.-X., Lin, X.-Z., Sun, W. & Ahmad, G. (2006), Clustering large scale of XML documents, *in* 'Proc. 1st Int. Conf. on Advances in Grid and Pervasive Computing', GPC'06, pp. 447–455.

Wernecke, J. (2008), *The KML Handbook: Geographic Visualization for the Web*, 1 edn, Addison-Wesley Professional.

Westbrook, J., Ito, N., Nakamura, H., Henrick, K. & Berman, H. M. (2005), 'PDBML: the representation of archival macromolecular structure data in XML', *Bioinformatics* **21**(7), 988–992.

Wilson, R., Cobb, M., McCreedy, F., Ladner, R., Olivier, D., Lovitt, T., Shaw, K., Petry, F. & Abdelguerfi, M. (2003), *XML Data Management*, Addison Wesley, chapter Geographical data interchange using XML-enabled technology within the GIDB system, pp. 354–374.

Xu, R. & Wunsch, I. (2005), 'Survey of clustering algorithms', *IEEE Trans. Neural Netw* **16**(3), 645–678.

Yang, R., Kalnis, P. & Tung, A. K. H. (2005), Similarity evaluation on tree-structured data, *in* 'Proc. ACM SIGMOD Int. Conf. Management of Data', SIGMOD'05, pp. 754–765.

Yang, Y., Guan, X. & You, J. (2002), CLOPE: a fast and effective clustering algorithm for transactional data, *in* 'Proc. 8th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.', pp. 682–687.

Yoon, J. P., Raghavan, V., Chakilam, V. & Kerschberg, L. (2001), 'BitCube: A three-dimensional bitmap indexing for XML documents', *J. Intell. Inf. Syst.* **17**(2-3), 241–254.

Zaki, M. J. (2002), Efficiently mining frequent trees in a forest, *in* 'Proc. 8th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.', pp. 71–80.

Zaki, M. J. & Aggarwal, C. C. (2006), 'XRules: An effective algorithm for structural classification of XML data', *Mach. Learn.* **62**(1-2), 137–170.

Zhang, K. & Shasha, D. (1989), 'Simple fast algorithms for the editing distance between trees and related problems', *SIAM J. Comput.* **18**(6), 1245–1262.

Zhao, Y. & Karypis, G. (2002), Evaluation of hierarchical clustering algorithms for document datasets, *in* 'Proc. 2002 ACM CIKM Int. Conf. on Information and Knowledge Management', CIKM'02, pp. 515–524.

Zheng, Y.-T., Li, Y., Zha, Z.-J. & Chua, T.-S. (2011), Mining travel patterns from GPS-tagged photos, *in* 'Proc. 17th Int. Multimedia Modeling Conf.', MMM'11, pp. 262–272.

Zhu, Y.-W., Ji, G.-L. & Sun, Q.-H. (2010), Clustering GML documents using maximal frequent induced subtrees, *in* 'Proc. 7th Int. Conf. on Fuzzy Systems and Knowledge Discovery', Vol. 5 of *FSKD'10*, pp. 2265–2269.