SINGIDUNUM UNIVERSITY

Department of Postgraduate Studies

DOCTORAL DISSERTATION

ARABIC XML DOCUMENTS: SUMMARIZING, MANAGING, AND SECURING

Mentor: **Prof. dr Mladen Veinović**

Candidate: **Hesham Elzentani**

Belgrade, 2016.

SAŽETAK

W3C je predložio XML kao standard koji se koristi u web aplikacijama, transakcijama, dokumentaciji, sistemima za upravljanje bazama podataka i razmenu informacija između sistema preko Interneta. XML omogućava čuvanje različitih podataka bez obzira na to kako će biti prikazani. XML se koristi za kreiranje, ažuriranje i upit baze podataka. Kreirati i pisati XML dokumente jasno i za čoveka čitljivo, kao što je za mašinu čitljivo, je lako, tako da je lako kreirati i aplikacije koje procesuiraju ove XML dokumente. Generalno, sve vrste informacija se mogu izraziti kao XML dokumenta.

U poslednjih nekoliko godina, rast sadržaja na arapskom jeziku i broja korisnika na Internetu je znatno povećan. Arapski je jezik širokog govornog područja sa više od 375 miliona ljudi koji govore arapski jezik i preko 155 miliona ljudi, ili preko četrdeset procenata ove populacije koja govori arapski jezik, koji koriste internet. To predstavlja oko pet odsto svih korisnika Interneta u svetu. Broj korisnika interneta koji govore arapski jezik porastao je za faktor šezdeset u poslednjih petnaest godina (2000-2015). Ovaj rast upotrebe nadmašio je rast sistema pretrage, sumarizaciju arapskog teksta (kao što su dokumenti i web stranice), procese upita i procesore prirodnih jezika.

Arapska reč ima različite oblike sintakse i morfologije sa različitim značenjima. Gramatički, dokumenti sadrže različite oblike reči, uključujući derivaciju reči. Ovo izaziva probleme u obradi teksta, sumarizaciji dokumenata i pronalaženju informacionih sistema. Osim toga, postoji visok stepen gubitka informacija tokom procesa upita, sumarizacije dokumenata i pretraživanja izvora, posebno kod velikih dokumenata, tako da je gubitak informacija direktno proporcionalan veličini dokumenata tokom ovih procesa.

Ova teza opisuje RAX sistem, dizajniran za rangiranje arapskih dokumenata u procesima pronalaženja informacija. Predloženo rešenje u osnovi zavisi od sličnosti sadržaja teksta. Model koji smo osmislili može da se koristi za dokumente koji se nalaze u različitim formatima i koji su pisani na arapskom jeziku. Zbog složene jezičkih semantike tog jezika predloženo rešenje koristi čisto statistički pristup. Dizajn i implementacija su zasnovane na postojećim okvirima obrade teksta i referentne gramatike arapskog jezika. Glavni fokus našeg istraživanja bio je evaluacija različitih mera sličnosti koje se koriste za klasifikaciju arapskih dokumenata iz različitih oblasti i različitih kategorija dokumenata na osnovu kriterijuma upita predviđenog od strane korisnika.

Dalje, teza će razmotriti upotrebu sistema zaštite na RAX sistemu. Ova upotreba je kombinacija između XML zaštite (XML digitalni potpis i XML enkripciju) i SOAP poruka kako bi se kreiralo tajno okruženje između krajnjeg korisnika i modela RAX sistema, kao i proučili napadi na sistem zaštite i protivmere.

Ključne reči: Tekst sličnosti mere, Tekst klasifikacija, proizvodstvo Arapski dokumenti, XML dokumenti, zaštita XML dokumenata.

ABSTRACT

W3C proposed the XML as standard that used in web applications, transactions, documentations, database management systems and to exchange information between systems over the Internet. XML allows storing different data regardless of how it will be displayed. XML has been used to create, update and query databases. Create and write clear human-readable XML documents as well as machine-readable are easy, so it's easy to create applications that process these XML documents. Generally, all kinds of information can be expressed as XML documents.

In recent years, the growth of Arabic content and numbers of users on the Internet has greatly increased. Arabic is a widely spoken language with more than 375 million speakers and over 155 million, or over forty percent of these Arabic-speaking people use the Internet. This represents nearly five percent of all the Internet users in the world. The number of Arabian speaking Internet users has grown by a factor of sixty in the last fifteen years (2000-2015). This growth in usage has outpaced the growth in information retrieval systems, summarization of Arabic text (such as documents and web pages), query processes and natural language processors.

The Arabic word has different forms of syntax and morphologies with different meanings. Grammatically, documents contain different forms of words including derivations. This causes problems in text processing, document summarization and information retrieval systems. Furthermore, there is a high level of information loss during the processes of querying, document summarizing and information retrieval, especially with large documents, as information loss is directly proportional to the size of documents during these processes.

This thesis describes an RAX System designed for ranking Arabic documents in information retrieval processes. The proposed solution basically depends on the similarity of textual content. The model we have designed can be used for documents stored in the different formats and written in Arabic language. Due the complex lingual semantics of this language the proposed solution uses a pure statistical approach. The design and implementation are based on existing text processing frameworks and referent Arabic grammar. The main focus of our research has been the evaluation of different similarity measures used for classifying Arabic documents from different domains and different document categories based on query criteria provided by the user.

Further, the thesis will study the security issues of the RAX system. These issues combined between XML security (XML digital signature and XML encryption) and the SOAP message to create a secret environment between an end user and the RAX system model as well as study the security attacks and countermeasures.

Keywords: Text similarity measures, Text classification, Processing Arabic documents, XML Documents, Securing XML Documents.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. dr Mladen Veinović for the continuous support of my Ph.D. study and related research, for his patience, motivation, and knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank Dr. Goran Šimić. Without his support, it would not be possible to conduct this research.

Last but not the least, I would like to thank my family for supporting me spiritually throughout writing this thesis and my life in general.

FIGURES

TABLES

LISTING

GLOSSARY

A

AES - Advanced Encryption Standard algorithm.

API - Application Program Interface.

C

CBC - Cipher Block Chaining.

CMWA - Cluster based Mobility and Energy Aware.

D

DBLP - Database and Logic Programming Bibliography.

DOM - Document Object Model.

DSA - Digital Signature Algorithm.

DTD - Document Type Definition.

E

ECDSA - The elliptic curve digital signature algorithm.

EXsum - Element-wise XML summarization.

H

HTML - Hyper Text Markup Language.

I

IDF - Inverse Document Frequency.

IR - Information Retrieval System.

J

JPEG - Joint Photographic Experts Group.

N

NLP - Natural Language Processing.

NXD - Native XML database.

P

PDF - Portable Document Format.

PKC - Public Key Cryptography.

PKCS - Public Key Cryptography Standards.

Q

QName - Qualified Name.

R

RSA - Rivest-Shamir-Adleman.

S

SAX - Simple API for XML.

SGML - Standard Generalized Markup Language.

SOA - Service Oriented Architecture.

SOAP - Simple Object Access Protocol.

SSL - Secure Sockets Layer.

T

TF - Term Frequency.

U

URI - Uniform Resource Identifier.

V

VSM - Vector Space Model.

W

W3C - Wide Web Consortium**.**

WSDL - Web Services Definition Language.

X

XDBMS - XML Database Management System.

XKMS - XML key management specification.

XML - EXtensible Markup Language.

XML-QL - XML - Query Language.

XPath - XML Path Language.

XPointer - XML Pointer language.

XQL - XML Query Language.

XQuery - XML Query.

XSchema - XML Schema.

XSD - XML Schema Definition.

XSLT - Extensible Stylesheet Language Transformations.

TABLE OF CONTENTS

1. INTRODUCTION TO XML

EXtensible Markup Language (XML) is a text-based markup language originated from Standard Generalized Markup Language (SGML) and later developed by the World Wide Web Consortium (W3C) organization to describe semi-structured data. XML is both human and machine readable language as shown in Listing 1.1. XML has been used to share and exchange information between computers and applications on the Internet, as well as query and/or update a database (XML database) [1][2][3][4][5]. Next points describe briefly the purpose of inventing XML:

1. Needs to exchange information between systems over the Internet.
2. Needs to a language that support unification and integration to a wide variety of applications.
3. Compatibility with other languages such as HTML and SGML.
4. Easy to create and write a clear human-readable XML documents as well as machine-readable, easy to create applications that process these XML documents (see Listing 1.1).

Next points describe briefly the benefits of XML:

1. The extensibility of XML allows user to create his own self-descriptive elements (sometimes called tags) that appropriate to his application.
2. XML allows to store different data regardless of how it will be displayed.
3. XML has simplified the usage and creation of HTML documents that used in huge Internet web sites.
4. XML has been used to create, update and query databases.
5. Style sheets can be merged with XML.
6. Generally, all kinds of information can be expressed as XML documents.

Next sub-sections will talk about XML documents such as syntax, processing, XML databases and queries … etc. in more details.

1.1. XML Syntax

Any XML document has semantic structure and written according to syntactic rules. Syntax rules to create XML document are [1][3][5]:

1. XML declaration, must in first line of XML document, such as `<?xml version="1.0" encoding="UTF-8"?>` in Listing 1.1, which has XML version and character encoding.
2. XML document is a tree that has one and only one root element. From Listing 1.1 the root element is `<books>`.
3. Element name is case-sensitive.
4. Every element must be closed, such as in Listing 1.1, the start tag is `<book>` and the close tag is `</book>`.
5. XML document is defined as a tree of elements as shown in figure 1.1, the depth of XML tree is unlimited and the elements can be repeated and must nested in correct order, such as in Listing 1.1 element `<book>` is repeated and closed before the root `<books>`.
6. Element attributes have names and values, the value must quoted `id` attribute in Listing 1.1(`<book id="1">`).
7. Less than (<), greater than (>), ampersand (&), double quotes ("), and single quote (') are entities (references) of XML, such as < and > are used to delimit the presence of elements names.
8. XML has used Namespaces to eliminate vocabulary restrictions. Namespace begins with `xmlns`:, namespace prefix and URI (Universal Resource Identifier), which's used as an attribute value in the element that assigned to a group of descendants elements, such as `<books xmlns:bk="www.library.com/books">`.

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book id="1">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
  <book id="2">
    <title>XML by Example</title>
    <author>Benoit Marchal</author>
  </book>
</books>
```

Listing 1.1. Example of XML document.

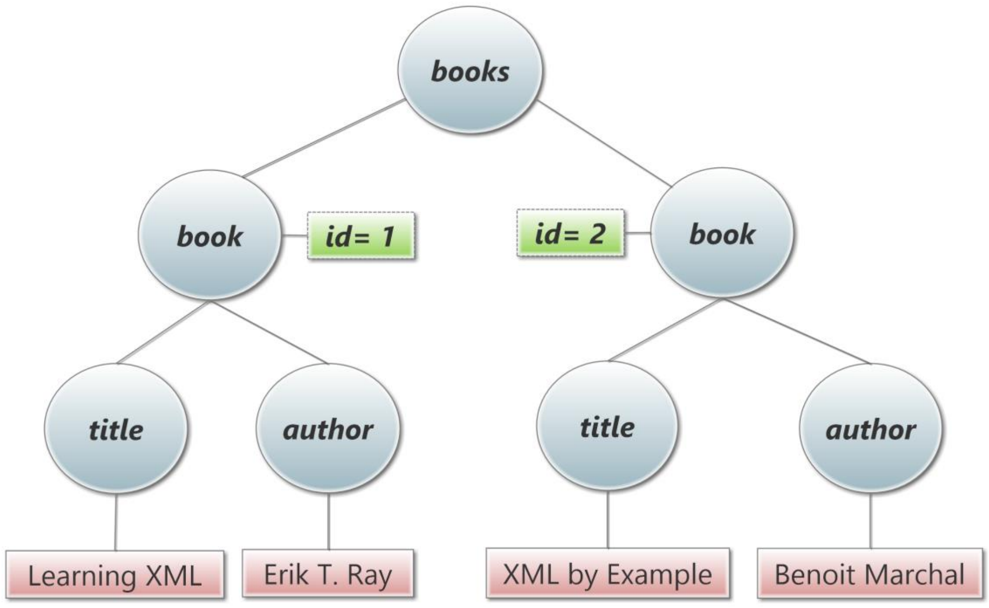


Figure 1.1. Tree of XML document in Listing 1.1.

1.2. XML Document Modeling

Based on XML features, you can create your own markup language, including elements and attributes definitions that suit for information you want to describe. The formal way to describe information as XML document is modeling, which deals with semantics of XML to determine the elements and their relationships in XML document, this thesis will discuss briefly two modeling tools[4][5]:

1. Document type definitions (DTDs) modeling tool, which describes the declaration rules and structures of documents.

2. XML Schema modeling tool, which uses templates to describe structure of documents.

1.2.1. Document Type Definitions

DTD can found in the beginning of an XML document or in a separated. DTD statements begin with a `DOCTYPE` declaration followed by the name of root element, list of all elements and attributes respectively, which describe the relationships in the XML document [3][4]. Listing 1.2 shows a DTD of XML document listed in listing 1.1.

```
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE books [
<!ELEMENT books (book+)>
<!ELEMENT book ( title )>
<!ELEMENT title ( #PCDATA )>
<!ELEMENT book ( author )>
<!ELEMENT author ( #PCDATA )>
<!ATTLIST book id CDATA #REQUIRED>
]>

<books>
  <book id="1">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
  <book id="2">
    <title>XML by Example</title>
    <author>Benoit Marchal</author>
  </book>
</books>
```

Listing 1.2. DTD of XML document in listing 1.1.

1.2.2. XML Schema

XML Schema (XSchema) or XML schema definition (XSD) used to describe the formal template to validate structure and content of XML document. XML schema is declared using the Namespace `<xs:schema xmlns:xs="http://www.w3.org /2001/XMLSchema">` followed with declaration of elements, attributes and data types [3][4]. Listing 1.3 illustrates the XSD of XML document listed in listing 1.1 and figure 1.2 shows graphical representation of it.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="books">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="book"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title"/>
        <xs:element ref="author"/>
      </xs:sequence>
      <xs:attribute name="id" use="required"
                    type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="author" type="xs:string"/>
</xs:schema>
```

Listing 1.3. XSD of XML document in Listing 1.1.

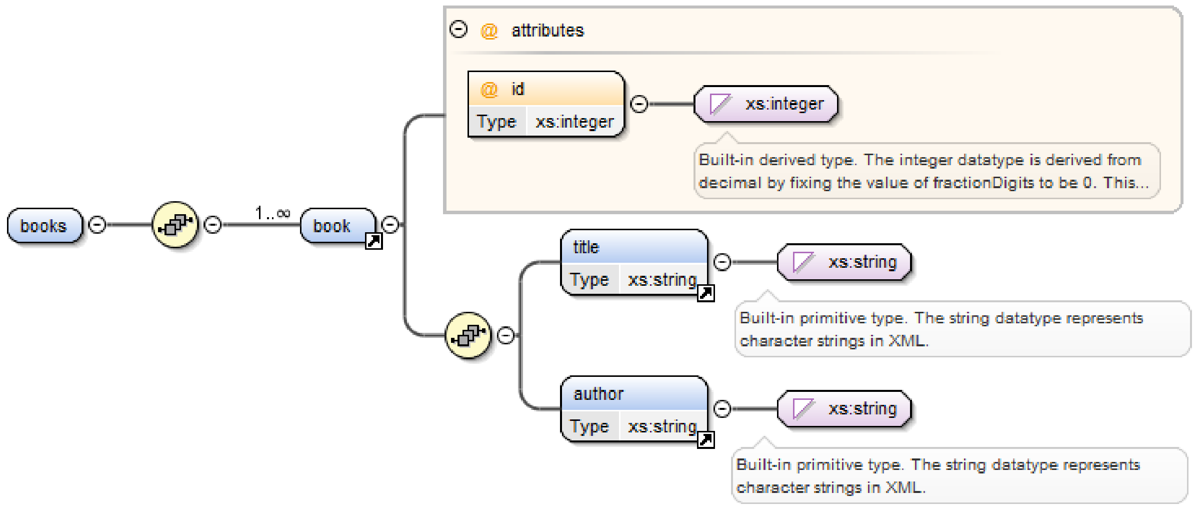

attributes
id
Type xs:integer
xs:integer
Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
books
1..∞
book
title
Type xs:string
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
author
Type xs:string
xs:string
Built-in primitive type. The string datatype represents character strings in XML.

Figure 1.2. Graphical representation of a schema listed in listing 1.3.

1.2.3. XML Document Validation

The validation process is done by XML parsers to validate XML document in two stages:

1. Well-formed XML document, and
2. Valid XML document.

XML document is well-formed, that’s mean it follows the XML syntax rules, (see XML syntax in section 1.1). The XML parser checks syntax rules in XML documents regardless to their structures (DTD or XSD), such as XML document listed in listing 1.1 is well-formed regardless to DTD or XSD, sometimes XML documents are well-formed but not valid [4][5].

XML parser checks if the document is syntactically right and has correct structure according to Schemas, so XML document is valid if it’s well-formed and has valid structure according to DTD or XSD, that’s mean all the elements, attributes and data of XML document follow the declaration of DTD or XSD, such as XML document in listing 1.2 and listing 1.3, which are valid based on DTD and XSD respectively [4][5].

1.2.4. Extensible Stylesheet Language Transformations

XSLT uses XML stylesheet language to transform an XML document into another different XML document as illustrated in figure 1.3, also XSLT extracts information from XML documents. The style sheet is well-formed XML document that illustrates the tree of input document, the tree of output document, and how transformation was made. In general XSLT has been used for following purposes [5]:

1. Add elements for styling, such as phone number or address to an XML contact document.
2. Create new table of contents from existing one.
3. Extract information from XML documents for readers.
4. Transform between XML documents that have different DTDs.
5. Transformation of XML documents into HTML documents.

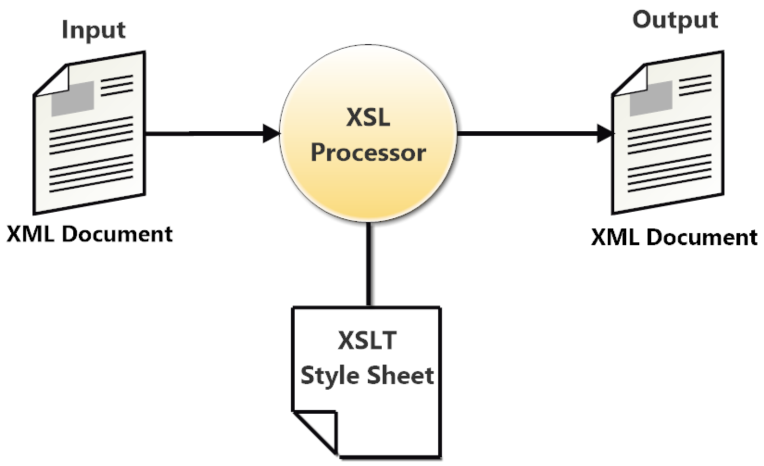


Figure 1.3. Transformation of XML document using XSLT.

1.3. XML Parser

Parsers are low level tools used by many applications to parse XML documents. The parsers are used in the as middle layer between applications and XML files. Figure 1.4 illustrates the framework of XML programs, which divided into two stages [5]:

1. Parsers used to communicate with XML files in low level, and
2. Applications that use data of XML file over parsers.

The parser reads the XML document as a tree and loads it into memory. This tree is exactly match the XML document and the applications process the tree as if it was XML document. In writing, the parser manipulates inserting and/or updating the tree through the applications, and finally the parser writes the updated tree into XML document.

The most famous methods used to communicate the parsers with applications are Object-based API and Event-based API. Document Object Model (DOM) and Simple API for XML (SAX) are interfaces used to analyze XML documents. The applications read the tree of XML documents through parsers concerning to syntax rules. Next sections will talk about DOM, which is Object-based interface and SAX, which is Event-based interface. In case of sharing, sending and receiving XML document over the Internet, the data needs to process by computers, this process called parsing that established DOM and SAX.

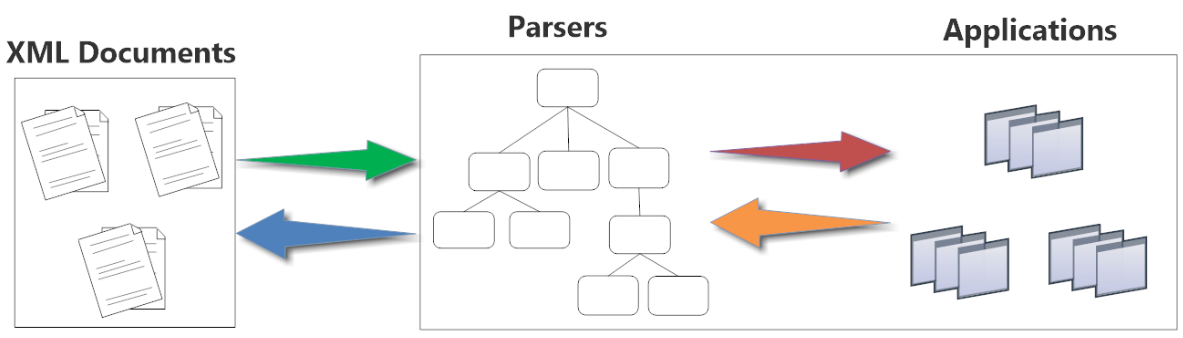


Figure 1.4. Framework of XML programs.

1.3.1. Document Object Model

DOM is developed by W3C as object-based API, DOM builds tree of an XML document according to syntax rules with all elements, attributes, Namespaces and data that document has, as well as DOM loads complete XML tree to memory to manipulate the tree through applications. XML tree has one root and the branches grow from this root (elements, sub elements, sub-sub elements and data). However, DOM tree consists of tree of nodes that are generic objects in the tree, figure 1.5 shows the content DOM node. To move freely forward or backward in the DOM tree, some properties must defined [1][3][4][5]:

1. Node name is the element or tag name.
2. Node type, every node has a code to represent the type of the object, as illustrated in table 1.1.
3. Parent node is the parent of current node.
4. Child node is the children nodes of current node.
5. First child is the first child of current node.
6. Last child is the last child of current node.
7. Previous sibling is the immediate previous node of current node.
8. Next sibling is the immediate following node of current node.
9. Attributes are the attributes of current node.
10. Node value is the value of the node, such as text data.

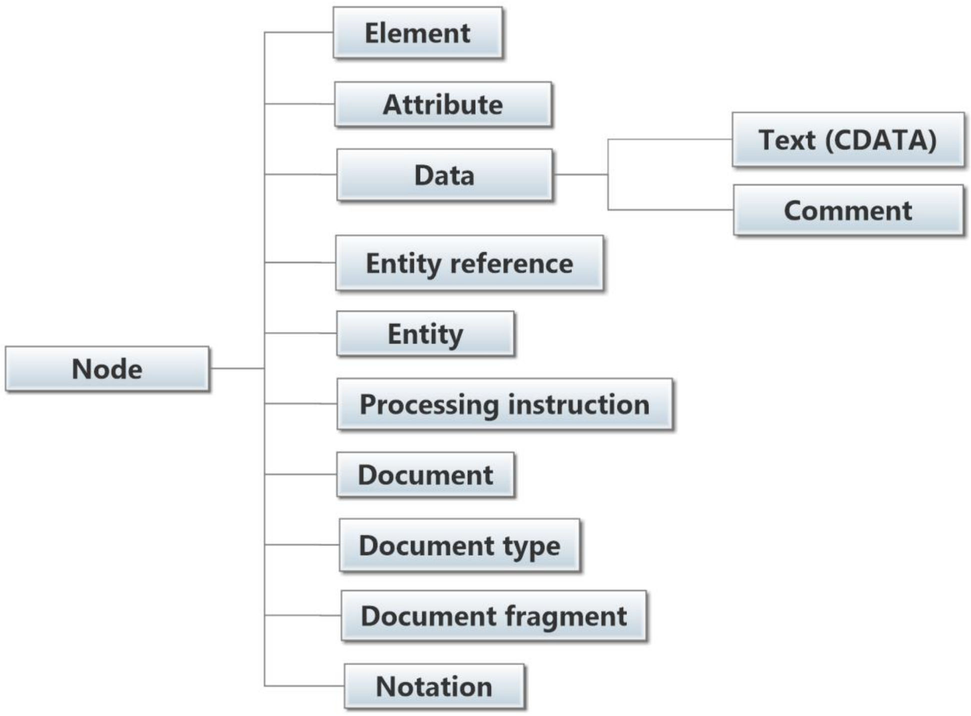


Figure 1.5. Content of DOM node.

Table 1.1. Code of node types.

Type	Code
Element	1
Attribute	2
Text	3
CDATA	4
Entity reference	5
Entity	6
Processing instruction	7
Comment	8
Document	9
Document type	10
Document fragment	11
Notation	12

The advantage of DOM parser is loading entire XML document to the memory as tree without losing any information before applications process the tree and the applications could move easily from one node to another. Disadvantage of DOM is memory consumption, big documents need a huge memory requirements and sometimes the system has failed.

1.3.2 Simple API for XML

SAX is event-based parser that reads XML document element by element from open tag to close tag or called event by event, SAX parser sends events as it reads them over the XML document and application reads and processes data immediately at that time. The events created for [1][5]:

1. Start document, event is triggered when XML document is opened.
2. End document, event is triggered when XML document is closed.
3. Element opening tag, event is triggered when start element is opened.
4. Element close tag, event is triggered when end element is closed.
5. Content of element, event is triggered after element is opened and before element is closed.
6. Entities.
7. Parsing errors.

SAX parser reads the document and generates an event, when SAX parser reads XML declaration will generate an event related to this declaration, when SAX parser reaches the first opening tag such as `<books>` will generate an event of element opening tag and notify the application that `books` element is started and so on.

SAX in advance is not memory intensive and its disadvantage is the difficulty of moving through tree data structure.

1.3.3 Comparison between DOM and SAX

DOM and SAX parsers complement to each other, DOM is the best in forming and editing, SAX is the best in exchanging information between applications.

However, figure 1.6 and table 1.2, illustrate the comparison between DOM and SAX parser based on application needs.

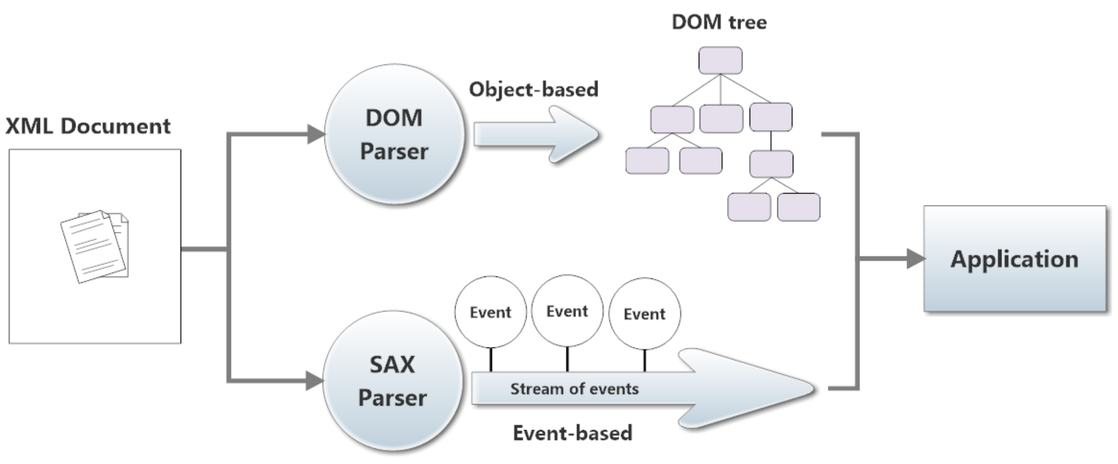


Figure 1.6. DOM and SAX comparison.

Table 1.2. Comparison between DOM and SAX.

Feature	DOM	SAX
Parser	DOM is object-based, builds a tree structure in memory from input XML document.	SAX is event-based, does not build any structure, SAX takes input XML document as events.
Processing	DOM processes entire XML document responding to any application request, no matter how much the application needs.	SAX processes the application request only with a part of XML document at any given time.
Speed and functions	DOM is the lowest in speed with a lot of powerful functions.	SAX is the fastest in speed with a fewer functions.
APIs type	Entire tree in memory	Stream of events
Use	To access different parts of XML document. When application has to change the tree many times and information has to be written in a specific amount of time.	When input XML document is too big for available memory. When only part of document is to be read. To implement less dynamic memory allocation.
XPath capability	Yes	No
Memory efficiency	Varies	Good
Movement	Forward and backward	Just forward
Read XML	Yes	Yes
Write XML	Yes	No
Insert, read, update and delete operations	Yes	No

1.4. XML Tree and Path

A tree *T* is a tuple $<r_T, N_T, E_T, \lambda_T>$, where $N_T \subseteq \mathbb{N}$ represents a set of nodes, $r_T \in N_T$ is the recognized root of *T*, $E_T \subseteq N_T \times N_T$ represents acyclic set of tree edges, and λ_T: $N_T \mapsto \sum$ is a function linking a node with a label in alphabet $\sum$.

Let *Tag*, *Att*, and *Str* be alphabets of tag names, attribute names and strings, respectively. An XML tree *XT* is a pair $XT = <T, \delta>$, such that:

1. *T* is a tree defined on the alphabet $\sum=$ *Tag-names* ∪ *Attribute-names* ∪ {*Strings*}, where *Strings* ∉ *Tag-names* ∪ *Attribute-names* denotes ***#PCDATA*** content.
2. Given $n \in N_T$, $\lambda_T(n) \in$ *Attribute-names* ∪ {*Strings*} ⇔ $n \in$ *Leaves(T)*,
3. δ: *Leaves(T)* ↦ *Strings* is a function linking a string to a leaf node of *T*.

An *XML path p* is a sequence of symbols in *Tag-names* ∪ *Attribute-names* ∪ {*Strings*}, where $p = s_1.s_2... .s_m$. Symbol s_1 matches the XML document root element. The types of XML path are:

1. *Tag path*, if $s_m \in$ *Tag-names*;
2. *Complete path*, if $s_m \in$ *Attribute-names* ∪ {*Strings*}.

P_{XT} denotes a set of complete paths in XML tree (*XT*), where $XT=<T, \delta>$, and *p* denotes XML path, where $p = s_1.s_2...s_m$. Applying *p* to *XT* will identify a set of nodes, where $p(XT)=\{n_1, ..., n_h\}$.

However, for each $I\in[1..h]$ there is a sequence of nodes, or *node path*, $\boldsymbol{np_i^p} = [\boldsymbol{n_{i_1}}, \dots, \boldsymbol{n_{i_{1m}}}]$ with following characteristics:

1. $\boldsymbol{n_{i_1}} = \boldsymbol{r_T}$ and $\boldsymbol{n_{i_m}} = \boldsymbol{n_i}$,
2. $\boldsymbol{n_{i_{j+1}}}$ is a child of $\boldsymbol{n_{i_j}}$, for each $j \in [1..\ m\text{-}1]$,
3. $\boldsymbol{\lambda}\left(\boldsymbol{n_{i_j}}\right) = \boldsymbol{s_j}$, for each $j \in [1..m]$.

Furthermore, a path that has been applied to an XML tree will return an *answer* based on the path type. So, when a tag path *p* is applied on *XT*, the answer is exactly the set of node that identifies *p(XT)*, which is $A_{XT}(p) \equiv p(XT)$, and when a complete path *p* is applied on *XT*, the answer defined as the set of string values linked to the leaf nodes identified by *p*, which is $A_{XT}(p) = \{\delta_T(n) \mid n \in p(XT)\}$ [6][7].

1.5. XML Tree Tuples

Tree tuples are the maximum number of subtrees that can derived from original XML tree using paths.

However, from the XML tree *XT*, tree tuples *T* is sub trees of *XT*, so for every tag path or complete path *p* in *XT* will return an answer $A_T(p)$ that contain at most one element, where T_{XT} denotes a set of tree tuples derived from *XT*.

Tree tuple represents a complete set of concepts that are linked to semantic structure of original XML tree.

Furthermore, tree tuples will maintain the same identical structure of original XML tree. In fact tree tuples present different ways to associate the content of original XML tree, based on XML document in listing 1.1 and its XML tree in figure 1.1, which represent two books, its noted that each internal node has unique element name and each leaf node has a name and value of an attribute, or symbol ***String*** and its value which's ***#PCDATA*** content.

Path answers can be easily founded, such as the path ***books.book.title*** returns the set of node identifiers $\{n_3, n_8\}$, whereas the path ***books.book.Author.String*** returns the set of strings $\{n_6, n_{11}\}$, which are {"Erik T. Ray", "Benoit Marchal"}. Two tree tuples can be derived from tree in figure 1.7, one tree tuple is derived starting from the left subtree rooted in the *books* element and the other tuple is derived starting from the right subtree rooted in *books* as shown in figure 1.8 [6][7].

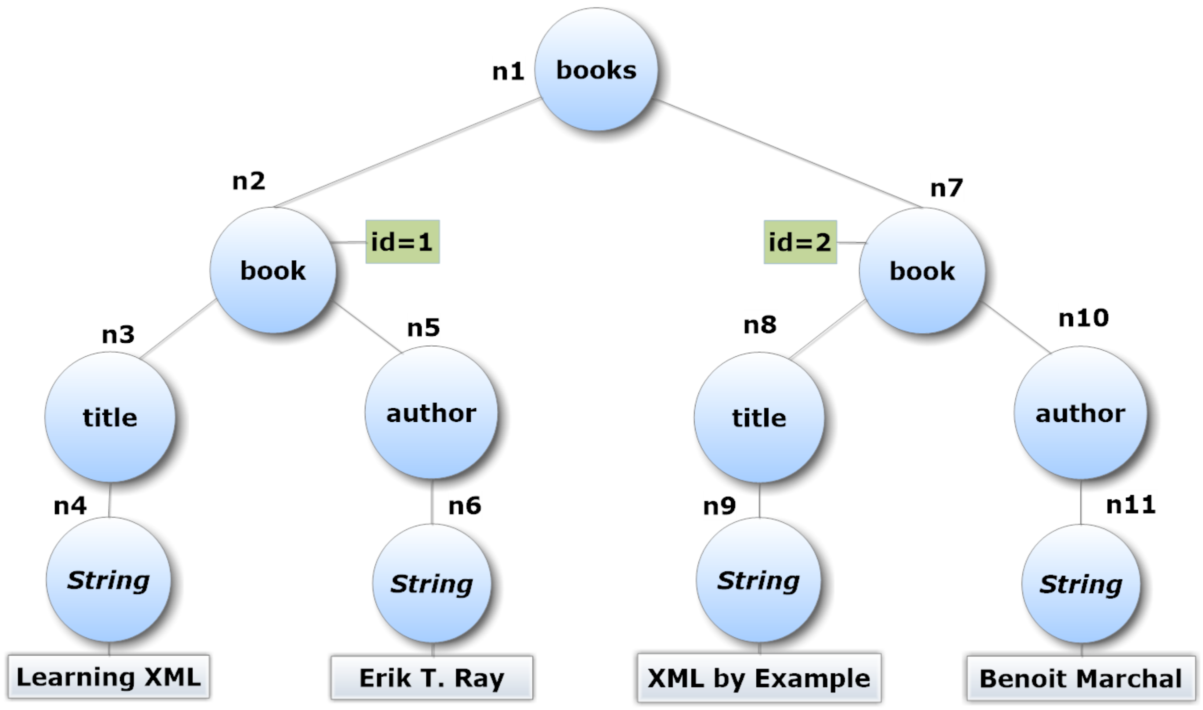


Figure 1.7. Nodes and XML tree of listing 1.1.

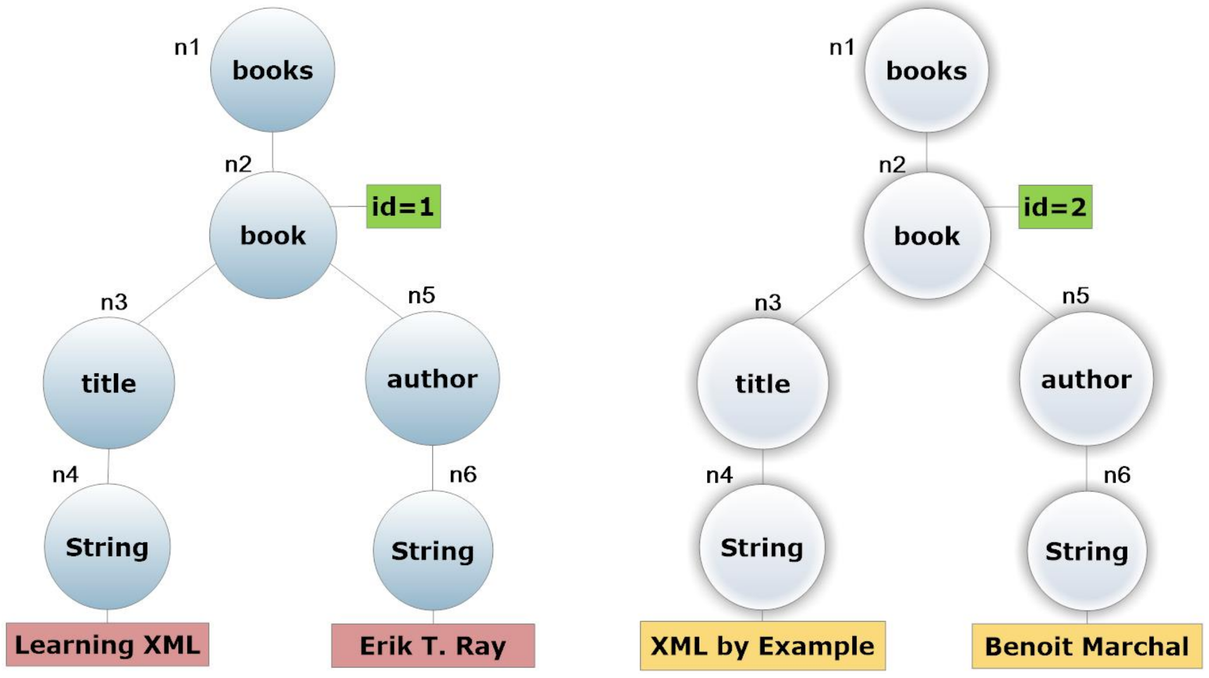


Figure 1.8. Tree tuples of XML tree in figure 1.7.

1.6. Data Addressing and Querying

Data Addressing and querying are the process where many query languages have been used to address and query semi-structured data such as XML documents, these query languages have a concept of path expressions to navigate XML tree, following is some samples of these query languages [8][9][10][11][12][13]:

1. XML-QL – XML query language used to extract parts of data from XML documents, as well as transformations, map and integrate of XML data between different DTDs from different sources.
2. XPointer – XML pointer, which's W3C recommendation that uses URI to point to specific pieces of XML document based on XPath expressions.
3. XQL – XML query language, that used to query XML databases.
4. XPath – XML path language, which's W3C recommendation describes paths to nodes and data in XML tree.
5. XQuery – XML query, which's W3C recommendation that uses XPath expressions to query XML data and relational databases.

The most used query languages are XPath and XQuery, so this thesis will discuss them in more details.

1.6.1. XPath Language

The XML Path Language is W3C recommendation, which's a query language used to select nodes and contents (node's data and values) of XML documents.

XPath is depending on tree representation of the XML document and has the ability to navigate over the XML tree as well as select nodes using some standards [14][15].

XPath navigates XML trees and return a set of matching nodes as an answer. In its simplest form, XPath expression looks like directory navigation paths, such as the XPath expression `/books/book/title` navigates from the root element (`books`), which proceeded by slash (`/`) through (`book`) elements, to their (`title`) child elements, so the result of entire XPath expression is a set of all (`title`) elements in the XML tree that can be amounted.

Moreover, at each step in the navigation way, the selected elements can be filtered using conditions. The conditions are boolean expressions written between brackets that used to test the existence of the paths, such as XPath expression `/books/book[author]` will return all `book` elements which have at least one `author` element as a child. Actually above XPath expression is a curtailment of explicit form `/child::book[child::author]`, that is mean in each path the query searches all the child element.

The same previous XPath expression can take another form, which is `/child::book/descendant::*[child::author]`, will return elements of any type that are descendants to the parent element `book` and have `author` sub-element.

Moreover, XPath expressions can use another axes like `preceding-sibling` to navigate backward through the elements that have same parent, or use `ancestor` to navigate upward frequently, to be clear see figure 1.9. The order of XML Document tree is defined as upward or backward axes depending on the type of navigation [16].

However, slash “/” in XPath expressions before an element refers to the root element and the element leads with no “/” refers to self-node in XML tree, which refers to the tree node where navigation starts from. So, when navigation starts form self-node in any XML tree, all other nodes can be reached based on XPath axes. XPath axes can partitioned XML tree from any self-node as shown in figure 1.9 as an example.

XPath expressions deals with a location paths, which selects a set of elements related to self-node location as example in figure 1.10 which selects two nodes of element `title`, so any location path can be expressed using following syntax [14][16]:

1. `child::book` – selects the children of `book` element of self-node.
2. `child::*` – selects all children element of self-node.
3. `child::text()` – selects all text of children node of self-node.
4. `child::node()` – selects all the children elements of self-node, no matter about node type.
5. `attribute::name` – selects the attribute name of self-node.
6. `attribute::*` – selects all attributes of self-node.
7. `descendant::book` – selects the descendants of `book` element of self-node.

8. `ancestor::book` – selects all ancestors of `book` element of self-node.

Some of XPath abbreviations used to express some cases [14][16]:

1. `*` – selects all elements of self-node.
2. `text()` – selects all text (String) node of self-node.
3. `@name` – selects the attribute name of self-node.
4. `@*` – selects all attributes of self-node.
5. `Last()` – selects the last child of self-node.

XPath operators are [12]:

1. Computes two node-sets (|).
2. Addition (+).
3. Subtraction (-).
4. Multiplication (*).
5. Division (div).
6. Equal (=).
7. Not equal (!=).
8. Less than (<).
9. Less than or equal to (<=).
10. Greater than (>).
11. Greater than or equal to (>=).
12. Or (or) .
13. And (and).
14. Modulus (mod).

XPath has more than 100 functions, the standard functions used in XPath are [12]:

1. Functions for string values.
2. Functions for numeric values.
3. Functions for date and time comparison.
4. Functions for node and QName manipulation.
5. Functions for sequence manipulation.
6. Functions for Boolean values.

For more details about XPath functions see [12].

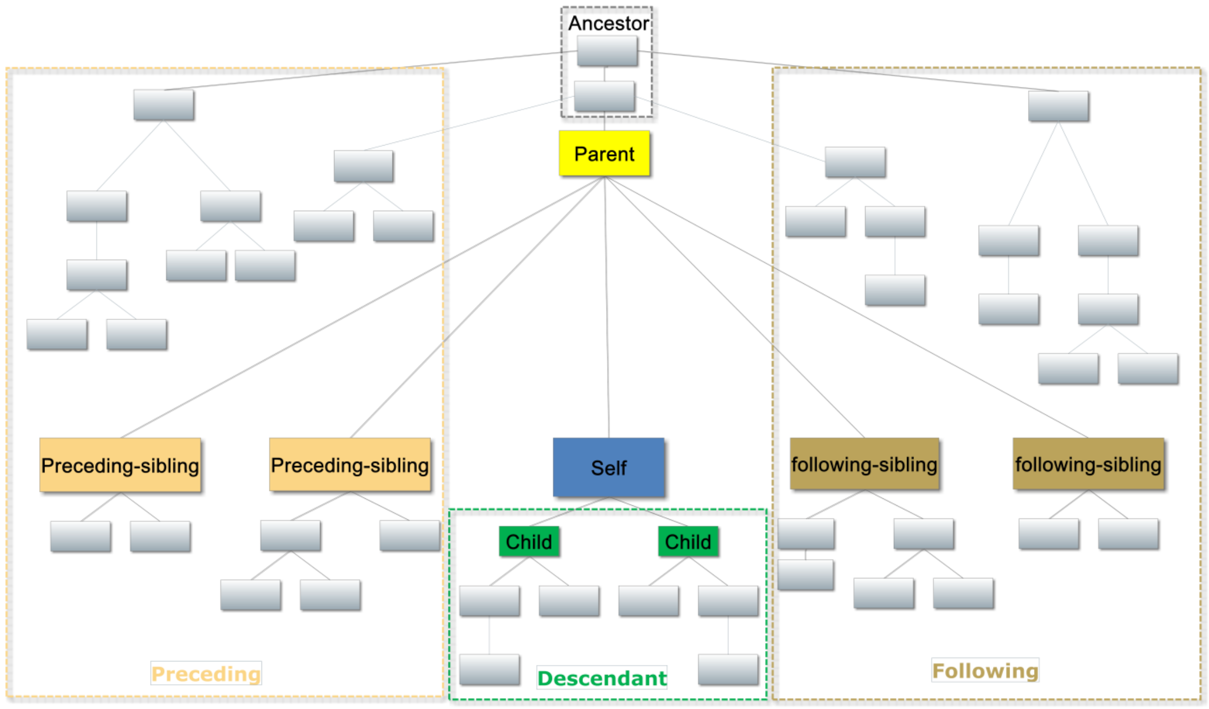
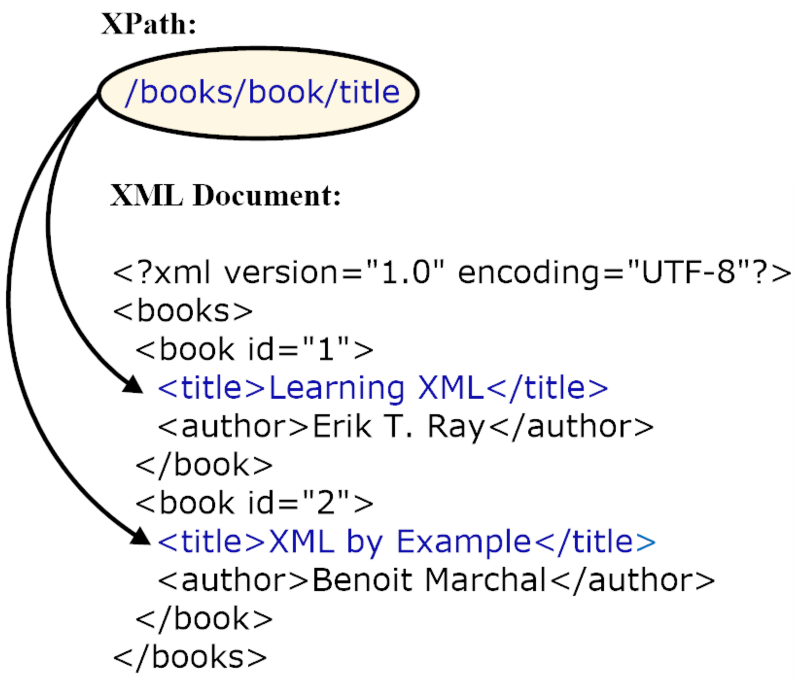


Figure 1.9. XPath Axes in XML tree.



```
/books/book/title
```

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
 <book id="1">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
 </book>
 <book id="2">
  <title>XML by Example</title>
  <author>Benoit Marchal</author>
 </book>
</books>
```

Figure 1.10. Example of location path.

1.6.2. XQuery Language

XQuery is a query language introduced by the W3C to manage semi-structured data like XML documents. XQuery has capability to make a lot of different types of operations on XML data such as [17]:

1. Select data according to a specific standard.
2. Data filtering.
3. Seeking for information within a document and/or collections of documents.
4. Joining data between documents or collections.
5. Sorting data according to a specific standard.
6. Grouping data according to a specific standard.
7. Aggregating data according to a specific standard.
8. Transformation of XML data into another XML data within different structure.
9. Making arithmetic operations.
10. Text processing.

XQuery uses syntax of XPath expression to address specific parts of an XML document based on FLWOR expression. The FLWOR expression can be used to establish the previous operations as well as grouping, aggregation and joining multiple data and documents. The FLWOR expression is built from clauses, the main clauses are [13][18]:

1. For clause,
2. Let clause,
3. Where clause,
4. Order by clause, and
5. Return clause.

Furthermore, listing 1.4 shows an example of XQuery used some FLWOR clauses and XPath expressions to return the `author` element as result of this query from XML document listed in listing 1.1.

```
<books>
{
  let $author:=doc('lib')//books/book/author
  return <book>{$author}</book>
}
</books>
```

Listing 1.4. Example of XQuery.

XQuery has been used to summarize and group XML documents to find out some results, such as average, summation or maximum values. There are a lot of functions built into XQuery that can used for [17]:

1. Process strings.
2. Process dates.
3. Establish mathematical calculations.
4. Combine sequences of tags.
5. Extra functions can be define according to special purposes.

Figure 1.11 shows how XQuery processor processing XML document and queries, where XML document as input may take different forms such as:

1. Text files as XML documents.
2. Fragments or parts of XML documents.
3. Collections of XML documents.
4. XML documents stored in XDBMS.

The XQuery processor used to parse and evaluate the XQuery expression as well as analyzing the syntax of query. The XQuery processor evaluates the result of XQuery expression according to the XML document (input document) and serialize the result to an XML document (output document) [17].

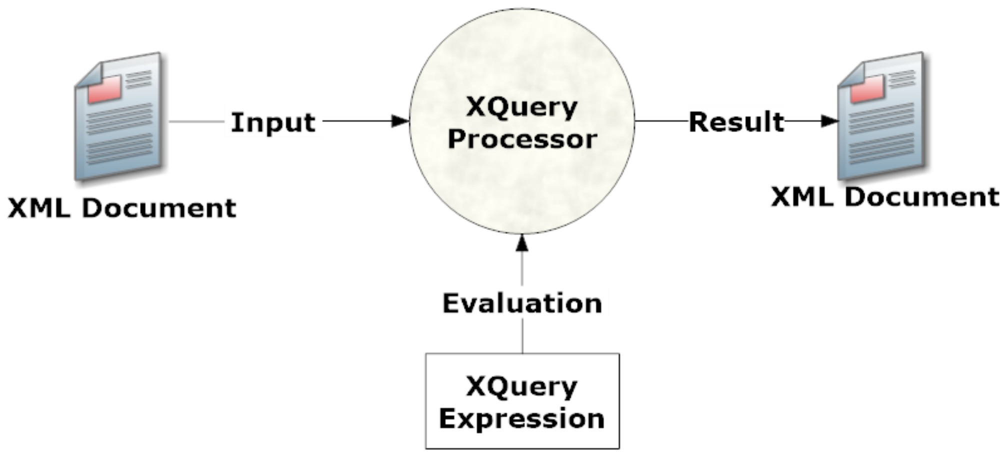


Figure 1.11. XQuery processor.

1.7 XML Database

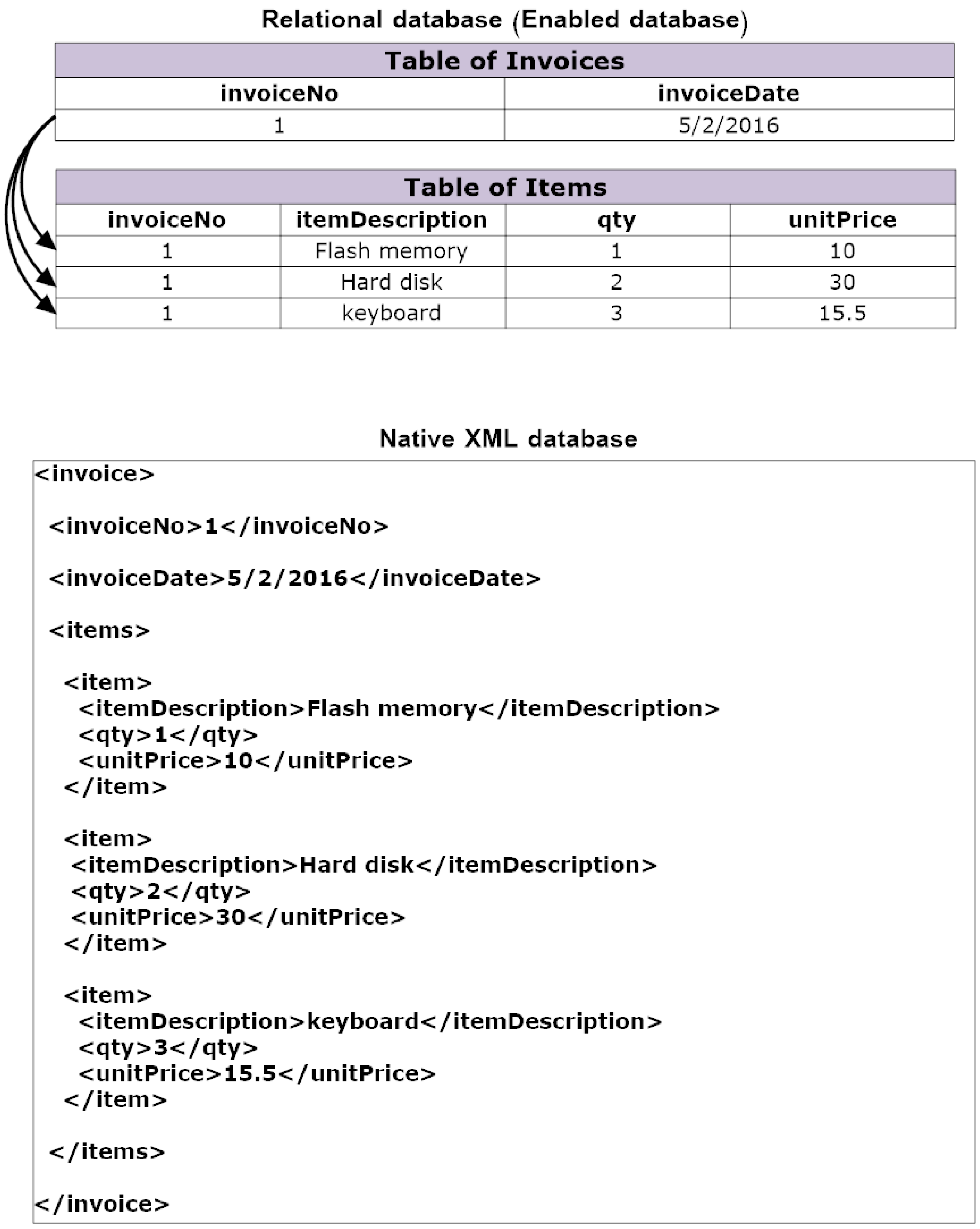
The XML databases are used to store huge data of XML format as a binary format, the XML database management system (XDBMS) loads documents to an application using DOM [5]. The data stored in XDBMS can be easily queried and manipulated using XPath or XQuery expressions. There are two main types of XML databases [3]:

1. Relational database, also known as enabled database, it stores data in set of records (tables of rows and columns) as shown in figure 1.12.
2. Native XML database (NXD), stores a huge data of XML documents and queried by XPath and XQuery expressions. It can store the same data of enabled database as shown in figure 1.12.

The advantage of NXD over relational database is the ability to store and query a huge data of XML documents than relational database, and the returned data is only in XML format is the disadvantage of NXD [3].

XML documents divided into two types, which are:

1. Data-centric documents, which used for data transport such as invoices and dynamic web pages.
2. Document-centric documents, which used XML language such as static web pages.



Relational database (Enabled database)

Table of Invoices

invoiceNo	invoiceDate
1	5/2/2016

Table of Items

invoiceNo	itemDescription	qty	unitPrice
1	Flash memory	1	10
1	Hard disk	2	30
1	keyboard	3	15.5

Native XML database

```
<invoice>

 <invoiceNo>1</invoiceNo>

 <invoiceDate>5/2/2016</invoiceDate>

 <items>

  <item>
   <itemDescription>Flash memory</itemDescription>
   <qty>1</qty>
   <unitPrice>10</unitPrice>
  </item>

  <item>
   <itemDescription>Hard disk</itemDescription>
   <qty>2</qty>
   <unitPrice>30</unitPrice>
  </item>

  <item>
   <itemDescription>keyboard</itemDescription>
   <qty>3</qty>
   <unitPrice>15.5</unitPrice>
  </item>

 </items>

</invoice>
```

Figure 1.12. Example of enabled and native databases.

In case of data-centric documents, which are well structured data, XML enabled database has been used. In case of document-centric documents, which are semi-structured data, NXD has been used [19].

In general, the reasons of using native XML database to store data as XML documents are [20]:

1. Data stored is semi-structured.
2. Retrieval speed is faster than relational database.
3. Easy to execute XML queries.

Furthermore, XDBMS is software used to store and query XML documents, such as MS SQL Server, Oracle DB, MarkLogic, Virtuoso and Sedna.

1.8. Securing XML Document

XML Document is known as a completely plain text document. However, on exchange data between applications, share information between computers over the web, electronic payments, electronic transactions, and querying databases can be easily spied, attacked and hacked by a third party. Therefore, W3C has developed standards to secure XML documents, which are XML Digital Signature, Encryption and XML key management specification.

Digital signature depends on PKC to grantee authentication, integrity and privacy. Furthermore, the signer after signing the document, he/she cannot deny the signature because his/her PKC has been used as well as any document alteration can cause invalid signature and the signature cannot be copied as well.

XML Digital Signature has three types; enveloped, enveloping and detached. XML Encryption has guaranteed completely secure XML document, after signing encryption will take a place using PKC of a recipient. The decryption process will take a place after exchange data has completed.

This thesis will talk about securing XML document in more details later.

2. ARABIC LANGUAGE

Arabic language is one of the sematic languages in the world. Arabic language is widely spoking language that has morphology, vocabulary and vowels. Arabic statement consists of (*Subject-Verb-Object*) or (*Verb-Subject-Object*) like others. The Arabic word is built by adding infix, prefix and/or suffix to the root as well as diacritics. The Arabic language has 28 letters, that written from *right-to-left*, in contrast to Latin which's written from *left-to-right*. The letters have many shapes based on their positions in the word. Arabic words are divided into nouns and verbs. Nouns include adjectives and adverbs, verbs include prepositions, pronouns and conjunctions. Nouns are masculine; feminine; singular, dual or plural as well as verbs, which are derived from roots [21].

In recent years, the Arabic language contents on the Internet (web pages, documents, … etc.) have grown as well as Arabic users have extremely increased, based on top ten languages in the Internet (table 2.1 and figure 2.1), Arabic is widely spoken language that exceeds 375 million speakers, there are 168,176,008 Arabic speaking people using the Internet, this represents 5 % of all the Internet users in the world; this means out of the estimated about 375 million persons in the world that speak Arabic, 44.8 % use the Internet. The number of Arabians speaking Internet users has grown 6,592.5 % in the last fifteen years (2000-2015). In contrast, this growth has faced with growth in information retrieval systems, summarization of Arabic text, such as documents, web pages, …etc., queries processes and natural language processors.

Table 2.1. Number of Internet Users by Language (http://www.internetworldstats.com/stats7.htm).

Top ten languages used in the web - November 30, 2015 (Number of Internet users by language)					
Top ten languages in the Internet	**Internet users by language**	**Internet penetration (% population)**	**Users growth in Internet (2000 - 2015)**	**Internet users % of world total (Participation)**	**World population for this language (2015 estimate)**
English	872,950,266	62.4 %	520.2 %	25.9 %	1,398,283,969
Chinese	704,484,396	50.4 %	2,080.9 %	20.9 %	1,398,335,970
Spanish	256,787,878	58.2 %	1,312.4 %	7.6 %	441,052,395
Arabic	**168,176,008**	**44.8 %**	**6,592.5 %**	**5.0 %**	**375,241,253**

Top ten languages used in the web - November 30, 2015 (Number of Internet users by language)					
Top ten languages in the Internet	**Internet users by language**	**Internet penetration (% population)**	**Users growth in Internet (2000 - 2015)**	**Internet users % of world total (Participation)**	**World population for this language (2015 estimate)**
Portuguese	131,903,391	50.1 %	1,641.1 %	3.9 %	263,260,385
Japanese	114,963,827	90.6 %	144.2 %	3.4 %	126,919,659
Russian	103,147,691	70.5 %	3,227.3 %	3.1 %	146,267,288
Malay	98,915,747	34.5 %	1,626.3 %	2.9 %	286,937,168
French	97,729,532	25.4 %	714.9 %	2.9 %	385,389,434
German	83,738,911	87.8 %	204.3 %	2.5 %	95,324,471
Top ten languages	2,632,248,147	53.5 %	787.0 %	78.2 %	4,917,011,992
Rest of languages	734,013,009	31.3 %	1,042.9 %	21.8 %	2,342,890,251
World total	3,366,261,156	46.4 %	832.5 %	100.0 %	7,259,902,243

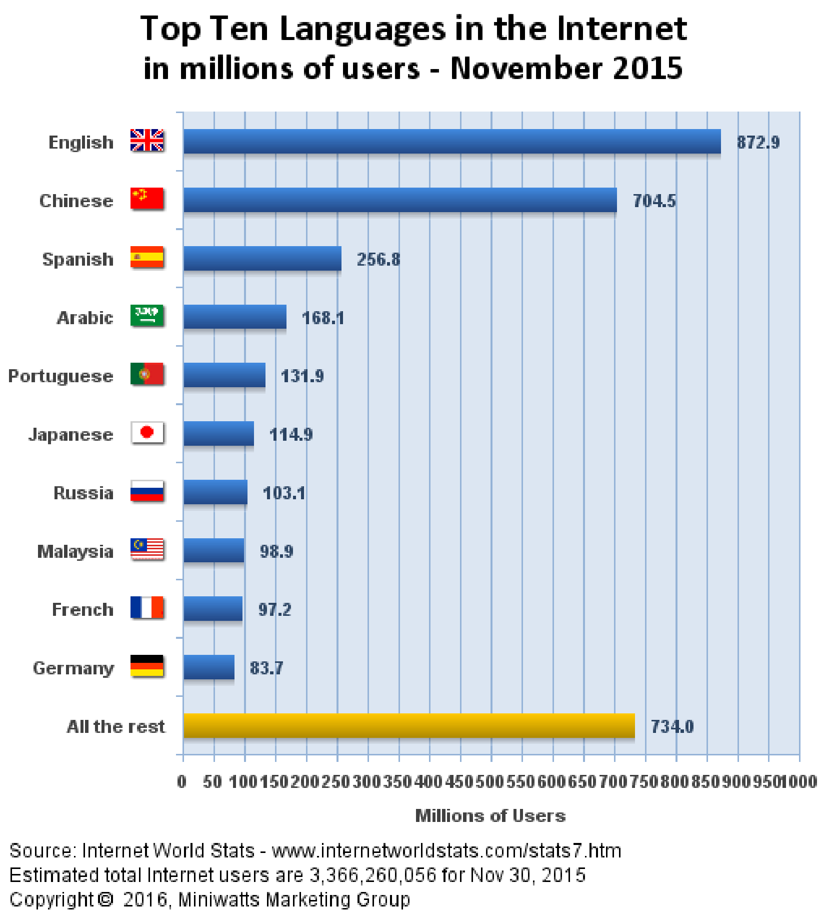


Figure 2.1. Top ten languages in the Internet - November 2105 (http://www.internetworldstats.com/stats7.htm).

2.1. Standard Arabic Reference Background

Arabic language has 28 letters, every letter changes his shape according to its position in the word. Table 2.2 shows Arabic letters and their shapes, figure 2.2 shows the structure of an Arabic word and an example of ligature [22].

Table 2.2. Arabic letters and their shapes.

Letter name	Independent	Beginning	Middle	End
Hamza (consonant)	ء			
Alif	ا	ا	ـا	ـا
Baa	ب	بـ	ـبـ	ـب
Taa	ت	تـ	ـتـ	ـت
Thaa	ث	ثـ	ـثـ	ـث
Jiim	ج	جـ	ـجـ	ـج
Haa	ح	حـ	ـحـ	ـح
Xaa	خ	خـ	ـخـ	ـخ
Daal	د	د	ـد	ـد
Dhaal	ذ	ذ	ـذ	ـذ
Raa	ر	ر	ـر	ـر
Zay	ز	ز	ـز	ـز
Siin	س	سـ	ـسـ	ـس
Shiin	ش	شـ	ـشـ	ـش
Saad	ص	صـ	ـصـ	ـص
Daad	ض	ضـ	ـضـ	ـض
Taa	ط	طـ	ـطـ	ـط
Zaa	ظ	ظـ	ـظـ	ـظ
Ayn	ع	عـ	ـعـ	ـع
Ghayn	غ	غـ	ـغـ	ـغ
Faa	ف	فـ	ـفـ	ـف
Qaaf	ق	قـ	ـقـ	ـق
Kaaf	ك	كـ	ـكـ	ـك

Letter name	Independent	Beginning	Middle	End
Laam	ل	لـ	ـلـ	ـل
Miim	م	مـ	ـمـ	ـم
Nuun	ن	نـ	ـنـ	ـن
Haa	ه	هـ	ـهـ	ـه
Waaw	و	و	ـو	ـو
Yaa	ي	يـ	ـيـ	ـي

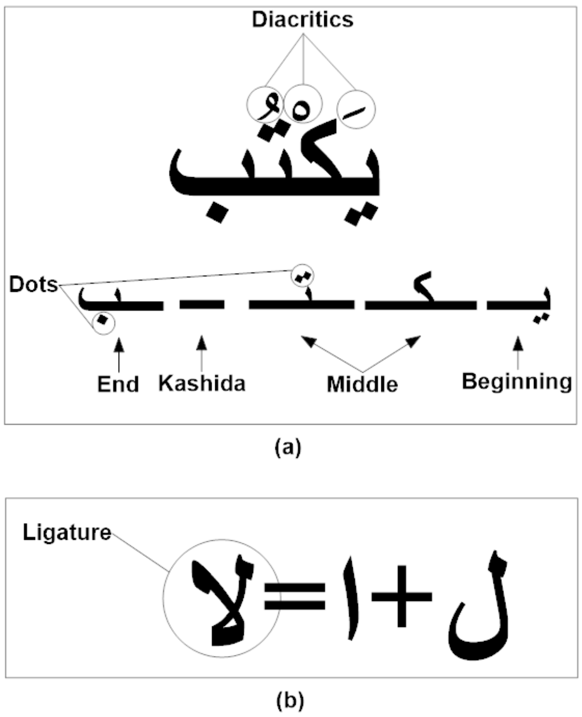


Figure 2.2. (a). Structure of an Arabic word with diacritics and letters shapes, (b). Example of ligature.

Arabic morphology or word structure deals with two main cases: lexical morphology or derivational (how words are built) and inflectional morphology (how words interact with syntax) such as genders, tenses, cases, and numbers. Arabic words has three types [21][22]:

1. Verbs, which are derived from roots.
2. Nouns, including adjectives and adverbs.
3. Particles, such as pronouns and prepositions.

Verbs are derived from linguistic roots of three, four, or five letters using affixes (suffixes, infixes, and prefixes), for example the verbs derived from the root كتب (ktb) are illustrated in table 2.3.

Table 2.3. Derived words from the root **كتب** (ktb).

Meaning	Pronunciation	Word
Book	Kitaab (n.)	كِتاب
Books	Kutub (n.)	كُتُب
He corresponded	Kaataba (v.)	كَاتَبَ
He is writing	Yaktub (v.)	يَكْتُب
He wrote	Kataba (v.)	كَتَبَ
Library	Maktaaba (n.)	مَكْتَبة
Office, desk	Maktab (n.)	مَكْتَب
Offices; desks	Makaatib (n.)	مَكاتِب
She is writing	Taktub (v.)	تَكْتُب
She wrote	Katabat (v.)	كَتَبَت
We write	Naktubu (v.)	نَكْتُب
Write (an order)	Uktub (v.)	أَكْتُب
Writer	Kaateb (n.)	كَاتِب
Writers	Kuttaab (n.)	كُتّاب
Writing	Kitaaba (n.)	كِتابة
Written	Maktuub (PP)	مَكْتوب

Furthermore, words or stems may have multiple affixes to build the words including conjunctions, propositions, pronouns, determiners and genders as well as suffixes used to structure plurals by number markers to form broken plurals [21].

Most words consists of a root and pattern connected together to build one word. Neither the root nor the pattern can exist separately, such as from table 2.3 the word Kaatib (*writer*) consists of two morphemes, which are the root k-t-b and the pattern _aa_i_ (where the underscores are places of the root consonants). So when the root and the pattern connected to each other, they build the word Kaatib (*writer*) [22].

Arabic language has masculine and feminine genders; singular, dual, and plural numbers; and nominative, genitive, and accusative grammatical cases.

A noun has the nominative case when it is a subject; accusative when it is the object of a verb; and genitive when it is the object of a preposition. Furthermore, the type of an Arabic noun is found by its gender, number, and grammatical case. Arabic nouns have a root, or stem, which is used in a word list or looked up in a dictionary. The nouns will be in the nominative case, such as خريطة (*map*). The definitive nouns are formed by attaching the Arabic article ال (*the*) to the immediate front of the nouns, such as the feminine Arabic word الخريطة (*the map*). Sometimes a preposition, such as ب (*by*) and ل (*to*), are attached to the beginning of a noun, usually in front of the definitive article as the masculine Arabic word بالخرائط (*by the maps*). Also a noun can carry a suffix which is often a possessive pronoun, such as the Arabic word بسيارتي (*by my car*) can be analyzed to ي + سيارة + ب, with one prefix ب (*by*) and one pronoun suffix ي (*my*) [22][23].

In Arabic language, the conjunction words such as و (*and*), إذا (*if*), من (*from*), لكن (*but*), كما (*as*), حتى (*even*) …etc. are often attached to the followed word, for example the conjunction word و (*and*) in the masculine word وبطالبها (*and by her student*).

Arabic has three kinds of plurals: sound plurals, broken plurals of certain patterns and human/nonhuman homonyms. The sound plurals are formed by adding plural suffixes to singular nouns. The plural suffix is ات for feminine nouns in all three grammatical cases, ون for masculine nouns in nominative case, and ين for masculine nouns in genitive and accusative cases. For example, the masculine word مدرّسون (*teachers*) is the plural form of masculine word مدرّس (*teacher*) in nominative case, and the masculine word مدرّسين (*teachers*) is the plural form of masculine word مدرّس (*teacher*) in genitive or accusative case. The plural form of feminine word مدرّسة (*teacher*) is the feminine word مدرّسات (*teachers*) in all three grammatical cases, here the dual suffix is ات for the nominative case, and ين for the genitive or accusative. The word مدرّسان (*two teachers*) is another kind of dual plural. The structure of broken plurals is more complex and often irregular; it is, therefore, difficult to predict. Furthermore, broken plurals are very common in Arabic such as, the plural form of the noun طفل (*child*) is أطفال (*children*), which is formed by attaching the prefix أ and inserting the infix ا. The plural form of the noun كتاب (*book*) is كتب (*books*), which is formed by deleting the infix ا. The plural form of امرأة (*woman*) is نساء (*women*). The plural form and the singular form are almost completely different [22][23].

Furthermore, the Arabic nouns have a lot of variants, as well as the complicated variants, because of the prefixes, suffixes, and infixes, such as the word ولأطفالها (*and to her children*) can be analyzed to ها + أطفال + ل + و , it has two prefixes and one suffix. Sometimes two nouns may look identical but they have different meanings, one human and one nonhuman, and the plural is different based on the noun referent as well, such as the word عمّال (*workers*) is plural of word عامل (*worker*) and the word عوامل (*factors*) is plural of word عامل (*factor*) [23].

The Arabic adjective can also have many variants, when the adjective modifies a noun in a noun statement, the adjective works in with the noun in gender, number, case and definiteness. The adjective has a masculine singular form such as جديد (*new*), a feminine singular form such as جديدة (*new*), a masculine plural form such as جُدُدّ (*new*), and a feminine plural form such as جديدات (*new*). For example, المدّرس الجديد (*the new teacher*) is masculine, and المدّرسون الجددّ (*the new teachers*) are masculine. The adjective has the feminine singular form when the plural noun denotes something inanimate. As an example, the word جديدة (*new*) in الكتب الجديدة (*the new books*) is the feminine singular form [23].

The Arabic verbs have two tenses, which are perfect and imperfect. Perfect tense denotes to complete actions and vice versa. The imperfect tense is jussive, indicative, imperative and subjective. Arabic verbs in perfect tense is composed of a root and a subject marker. However, the subject marker denotes to the number of the subject, person and gender. Although, the shape of a verb in perfect tense can have subject marker and pronoun suffix. The form of a subject-marker is specified together by the person, gender, and number of the subject. For example دَرَسَ (*to study*) is the perfect tense of دَرَسَتْ for the third person, feminine, singular subject; دَرَسُوا for the third person, masculine, plural subject. A verb with subject marker and pronoun suffix can be a complete sentence. For example, the word دَرَّسَتْهُ has a third-person, feminine, singular subject-marker ت (*she*) and a pronoun suffix ه (*him*), it’s also a complete sentence, meaning “*she taught him*”. Often the subject-makers are suffixes, but sometimes a subject-marker can be a combination of a prefix and a suffix, such as the word go back in a negative sentence is لا ترجعي (*Don’t go back*). The verbs in imperfect tense, in addition to the subject-marker can also have a mood-marker [22][23].

2.2. Stemming

Stemming a word is the opposite of the construction process which’s built the word. Stemming process used in linguistic morphology and information retrieval systems to reduce morphological variations of words (derivations) to a common stem or root.

There are two types of stemming:

1. Dictionary-based, which uses lists of related dictionary words.
2. Algorithmic-based, which uses programs to define related words.

Stemming is actually established by removing affixes from the word before final assignment of the word to the stem. Further, the stem of the word illustrates wider notion than the original word, so in IR system, the process of stemming increases the number of retrieved documents. Documents grouping and summarization process need stemming process as well before go further in processing [24].

2.2.1. Arabic Stemmers

Arabic words are difficult to stem, so there are additional approaches to stemming Arabic words in addition to affix removal, which are morphological analysis approach, and statistical methods.

A. Manual Stemmer

Manually constructed dictionaries are manually created the dictionaries of roots and stems for each indexed word [26].

B. Affix Removal Stemmer

First step in preprocessing is normalization, which established by:

1. Transforming Arabic encoding to UTF-8.
2. Delete punctuations.
3. Delete diacritics and kashidas.
4. Stripping of none letters and numbers.
5. Replace إ ،أ, and آ with ا, replace ى with ي, and replace ة with ه.

The affix removal approach or light stemming, removes a limit set of prefixes and/or suffixes, but in contras it keeps infixes. The prefixes, such as (لل - ل - ا - و - س - ب - ي - ن - م - ت - ف), and suffixes, such as (هما - تما - كما - ان - ها - و- ا - تم - كم - تن - كن - نا - تا - ما -ون - ين - هن - هم - ته - تي - ني - ن - ك - ه - ة - ت - ي - ات). Light stemmer removes frequently strings that found as prefixes or suffixes. In contrast, light stemmer doesn't remove infrequently strings that found at the beginning or ending of stems. Light stemmer can stem correctly many variants of Arabic words into their roots or stems, but also fail to stem other Arabic words, such as broken plurals of nouns and adjectives do not get stemmed with their singular forms [25]. After normalization and removing stopwords are established, if the word consists of at least two morphemes then will stem it to a root of three consonants using word pattern. The conflation will depends on the query of a user and the document or collection of documents [26].

C. Morphological Analyzer

Morphology is a linguistics branch that study, descript, identify and analyze the morphemes which form a word, as well as affixation, roots and pattern.

Morphological analysis is the process that categorize and build a representative structure of the morphemes. There are two rules of word's morphemes [27]:

1. Orthographic rules, and
2. Morphological rules.

However, morphological analysis and lemmatization are NLP that analyze a text within internal structure of the words, such as part-of-speech tagger which used to analyze a given context, morphological characteristics (case, gender, number, person, tense, mood, voice, etc) and Google's search facilities [27].

Arabic morphological analysis is the basis of Arabic NLP systems that used in many software applications to find the root or any potential root, such as machine translation, information retrieval systems, lemmatization, text categorization, dictionary automation, spell checking, summarization and translation [26][27].

D. Statistical Stemmer

Statistical methods have widely been used in automatic morphological analysis in computational linguistics. In statistical methods the words are clustered or grouped according to different string similarity measures using n-grams. Some methods use co-occurrence analysis as statistical stemming method to build stems [26][27].

3. SUMMARIZATION

Summarization is based on NLP and machine learning technologies to summarize text. Summarization is a process that reduce the text to point important information of the original text.

The goal of summarization is to represent subset of data that contains information of whole set, such as document summarization and search engines like Google.

3.1. Introduction to Summarization

In a document summarization or multi-document (multiple source of documents) summarization, abstract of entire document has been created to describe the most informative information or sentences and even words.

Information has extremely grown on the Internet and increased to become a huge data with time, so the needs to develop automatic summarization technologies also increases. There are two approaches of automatic summarization (figure 3.1) as followed [28]:

1. Extraction-based, and
2. Abstraction-based.

Extraction-based summarization selects a subset of a document (phrases, sentences and words) to build the summarization of the document, such as automatic keywords extraction. In contrast, abstraction-based summarization uses language semantics and NLP techniques to build the summarization of the document.

In abstraction-based, the summary may contains words not found in the original document unlike extraction-based.

Extraction-based summarization has two types (figure 3.1), which are:

1. Generic summarization, and
2. Query-based summarization.

Generic summarization builds generic abstract or summary of collections, such as the collection of documents.

Query-based summarization builds abstract or summary of collections based on a query [28].

Furthermore, Summarization has four techniques (figure 3.1), which are [29]:

1. Rule-based technique (semantic and syntactic), such as NLP,
2. Clustering, which's grouping different documents based on their properties,
3. Statistical technique, such as binomial distribution, and
4. Machine learning technique, such as log linear model.

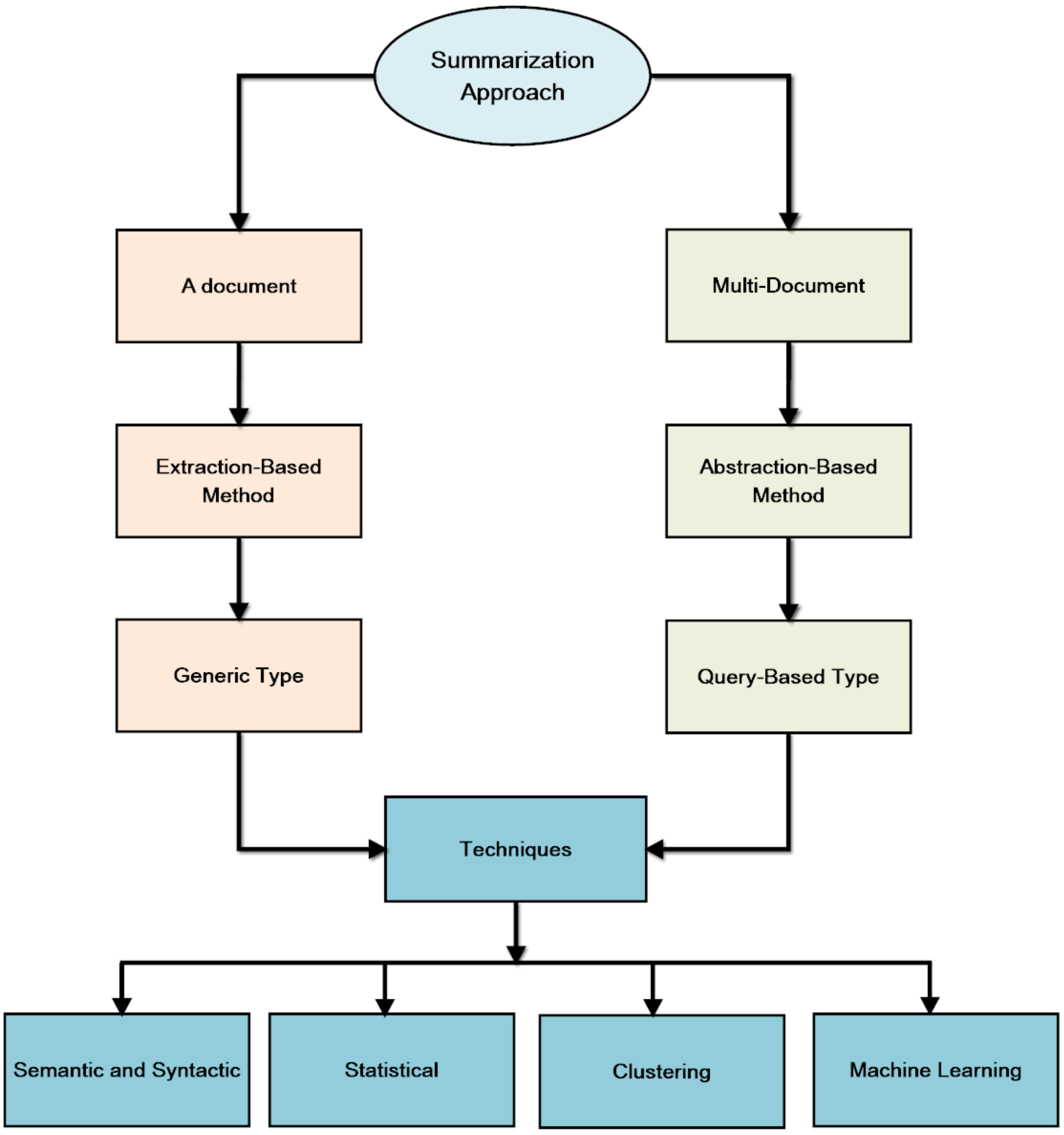


Figure 3.1. Summarization approach.

3.2. XML Document Summarization

Updating XML documents cause extreme increasing in size of documents with time and become a multi gigabytes with a lot of redundant information. This growth needs more storage memory, and more powerful capability processors to process the XML documents. So, there is a need to develop and build a reliable mechanism to access XML documents in local or remote computers and networks easily, such as summarize XML documents to meet the requirements of exchanging data, querying, sorting, comparing, or to be human readable more than machine readable, and to reduce the size of documents as well as to prevent memory consuming; taking in account the information importance of XML document as well as query importance, which belong to the part of XML document.

Although, much time has spent in reading and realizing the complex structure with large data of the XML documents. On the other hand, in a lot of cases, it’s impossible to read such documents. So its mandatory to present a summarized document of the original XML document, which has complex structure and large data. The summarized XML documents have been used by some applications, such as querying, sorting, comparing, some devices which have limited memory and limited processing abilities and storage spaces.

The challenge here is how to generate a perfect summarized document that helpful for users and applications as well.

Summarized XML document evaluated by three standards, which are document size, information content and information importance as following [30]:

1. Document size: size of summarized XML document obtained by applications, based on some algorithms and approaches. In general, the original documents with small size has good summarization and it can be easy readable by users and developers. In contrast, the original documents with large size; its summarization depending on XML structure and XML data; it has some difficulty in reading and realization by users and developers. The summarization ratio will calculated between the size of original document (in bytes) and the size of summarized document (in bytes) based on following equation: $Ratio_{summarization} = \frac{Size_{summarized}}{Size_{orginal}}$.

2. Information content: theoretically, a perfect summarized document contains the whole important information content of the original document. Practically, it's impossible to summarize an XML document that has not redundant information without losing some information content of the original one. The information content ratio will calculated between the number of element values of original document content without redundant information and the number of element values of summarized document content based on following equation: $Ratio_{content} = \frac{Content_{summarized}}{Content_{orginal}}$.
3. Information importance: as mentioned above, the summarized document cannot contains the whole important information of the original document. So, it's mandatory that the summarized document contains the most important information. Generated summaries used to summarize important information, such as generic-based summary which summarize the entire content of the document and query-based summary which summarize some parts of the document which are relevant to the user's query.

Furthermore, the summarization process needs to develop XML schema summarization technology, XML structure summarization technology, XML statistics summarization technology and querying XML database systems approaches, which concern about ranking the resulted XML documents as well as comparing similarity with query.

3.2.1. XML Schema Summarization Technology

Extracting well-formed XML schema form an existing XML document and selecting the important nodes in schema depends on an adaption algorithm and methods. So XSD and XML document summarization have three stages (figure 3.2) [31]:

1. *Schema extraction.* Most of XML documents found in the web are without XSD. So the approach in this step is to extract XSD form the source.
2. *Schema summarization.* In this step the extracted XSD will analyzed and summarized to create XSD summary that contains a proper size and the most important nodes.
3. *Document summarization.* Based on the relationship between XSD and its summary, an instance of XSD summery is obtained. So, the XML document of original schema will transformed to build the summarized XML document according to summarized schema.

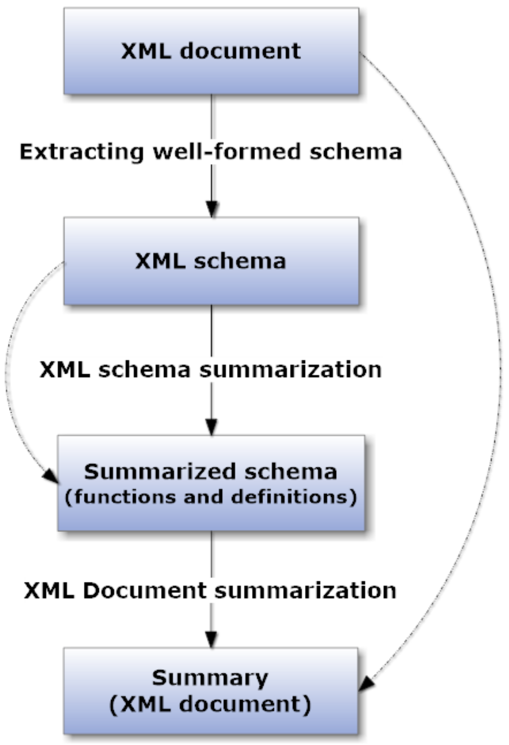


Figure 3.2. Schema summarization process.

Furthermore, schema summarization is a process where the most important and representative nodes are selected from the given schema to build another schema (schema summary). The user's needs control the size of schema summary. So, the size is determined by the number of nodes that schema summary has.

To create an XSD summary, the following information that provided by source XSD and its XML document are needed [31][32]:

1. *Source XSD*: XSD is a labeled directed graph, $S=<L, E_L, p, k>$, where: L is a finite set of labels; E_L is a finite set of directed edges between labels, $(l_1 \rightarrow 1_2) \in E_L$; p is the root label, $p \in L$; and k is a function that assign keys to labels, $k{:}L \rightarrow 2^L$.
2. *XML document*: XML document as instance is a labeled directed graph too, $I=<N, E_N, r, \lambda, v>$, where: N is a finite set of nodes; E_N is a finite set of directed edges between nodes, $(n_1 \rightarrow n_2) \in E_N$, where n_1 is parent of n_2; r is the root node, $r \in N$; λ is a function that assigns labels from L to nodes (the label l, such as $l=\lambda(n)$, is the type of n); and v is a function that assign text values to nodes.

3. A key for a label l is a set of labels (l_1, …, l_m), such as for any tuple of values, ($v(n_1)$, …, $v(n_m)$), $\lambda(n_i)=l_i$, there exist at most one subtree of type l in any instance of XSD.
4. The instance I must satisfy structural constraints and key dependencies to be an instance of an XSD S. So, $\lambda(r)=p$, and if $(n_1 \rightarrow n_2) \in E_N$, then $(\lambda(n_1) \rightarrow \lambda(n_2)) \in E_L$.
5. Algorithms used to determine the most important nodes in a given XSD, such as PageRank algorithm. The algorithm returns the weight of each edge in the XSD. Further, the weights are normalized to find the importance of XSD labels based on labels connectivity in the XSD and nodes cardinality in the instance. A relative cardinality $R(l_j \rightarrow l_k)$ of an edge $l_j \rightarrow l_k$ is the average number of nodes of l_j connected to nodes of l_k, where $R(l_j \rightarrow l_k)$, l_j and $l_k \in L$.
6. A schema summary SS=< L', E'_L, p, k > is a pair of (S, R_c), where R_c is a finite set of relative edges according to relative cardinality calculations, and S=<L, E_L, p, k> is the source schema, where $L' \subseteq L$; an edge $(l' \rightarrow l) \in$ E, if found a connection between l' and l in S; SS and S have the same root (p); and k is defined only for $l' \in L$.

3.2.2. XML Structure Summarization Technology

A structural summary of a document tree D is a summarized data structure from which specific structural properties of D can be concluded without access to D itself. Structural summaries have two types, which are centralized and decentralized. The centralized summary of D is a tree that has information about the set T of tags and nodes and their relations in D. The decentralized summaries include labelling schemes that identify an individual node and its tree relations in D using the local information of this node [33].

XML document has two structural properties, which are [34]:

1. Entity nesting, and
2. One-to-many relationships.

The structural properties are determined by instances of document, no need to DTDs or XSD. First of all, analyze the nesting of entities in an instance of an XML document. The star edge denotes to one-to-many relationship between two entities. There are some recursive entities the instance, such as DTD and XSD proof that a lot of recursive entities in the XML documents as well as relational databases. Then, analyze each one-to-many relationship, such as $A \rightarrow B^*$, where A and B are nodes in the instance. Finally, statistics used to describe the distributions stored in a prefix tree. A node in a prefix tree is linked with support nodes of the prefix in the instance. A support ratio calculated for each parent-child nodes (A, B) in the prefix tree to estimate the location of one-to-many relationship as following [34]:

1. The ratio is between 0 to 1, means B is probably child of A;
2. The ratio is equal to 1, denotes to one-to-one relationship;
3. The ratio is greater than 1, denotes to one-to-many relationship, and the edges known as star edges.

Furthermore, as an example of structural summarization, figure 3.3 shows a simple DBLP XML document and its prefix tree with support nodes (in square brackets), support ratio next to the edges and asterisk sign, which denotes to one-to-many relationship.

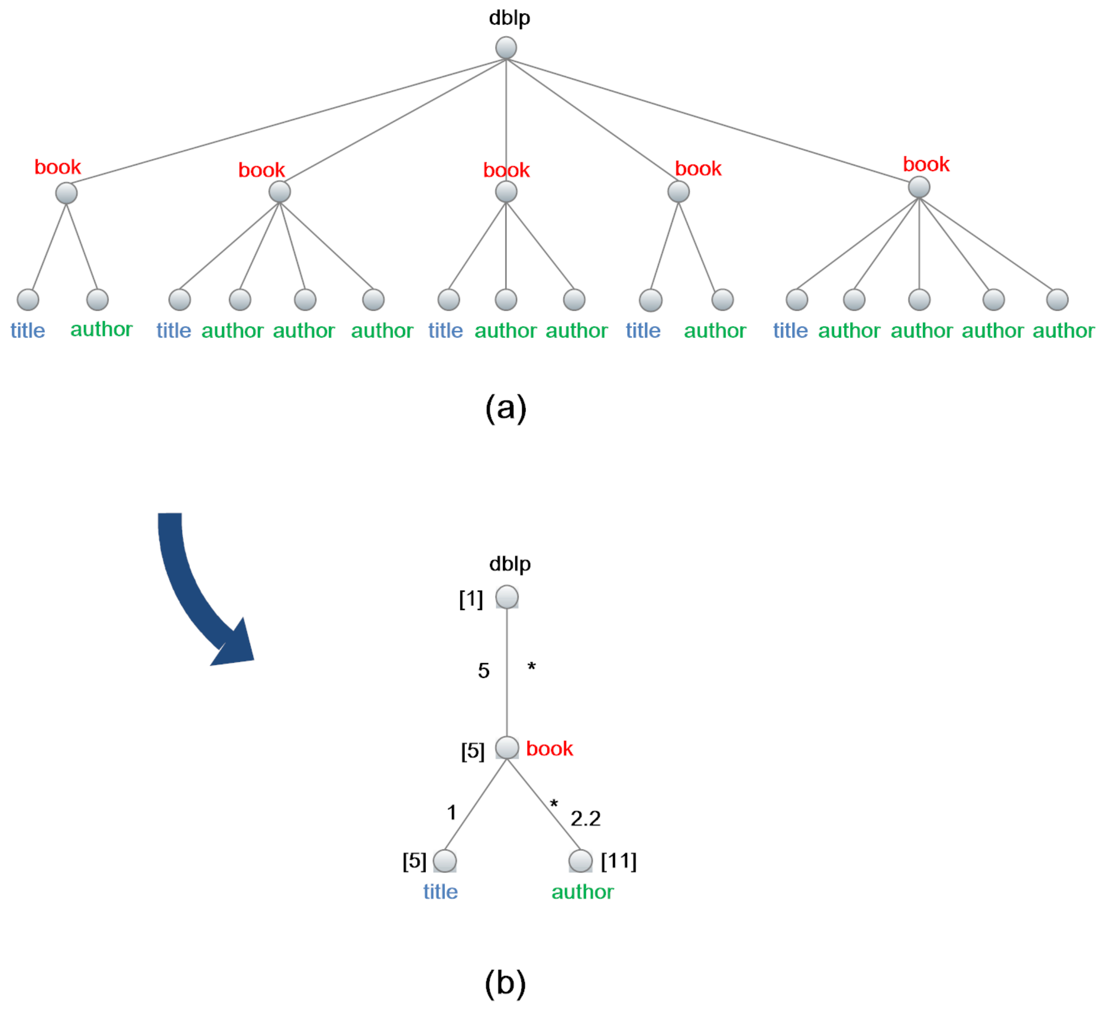


Figure 3.3. Structural summarization, (a). Simple DBLP XML document, (b). Its prefix tree.

3.2.3. Querying XDBMS Approaches

A result of any XML query is another XML document (XML fragment with hierarchical structure and value content) constructed from an original XML document in database system based on query structure.

XQuery is a powerful language that used widely as query language for XML database processes. The XML summarization is obtained by XQuery transformation from the original XML document. When a user send an query over an XML document or a collection, the query is transformed by XQuery transformation into a query to the XDBMS, finally the answer is approximated and returned to the user as summarized XML document (figure 3.4). If the structure of XML data is well known, queries always return precise answer, because each marginal value denotes to the exact value of the element and the frequency of the elements denote to the exact count of this value in the whole XML document, collection or XDBMS as shown in figure 3.5. Coverage and answer quality are factors used to analyze and evaluate the degree of summarization in querying approach [35].

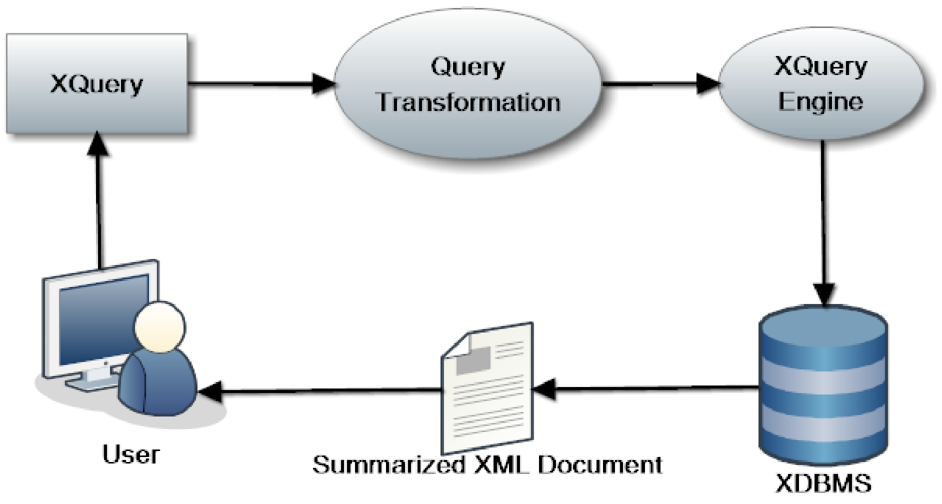
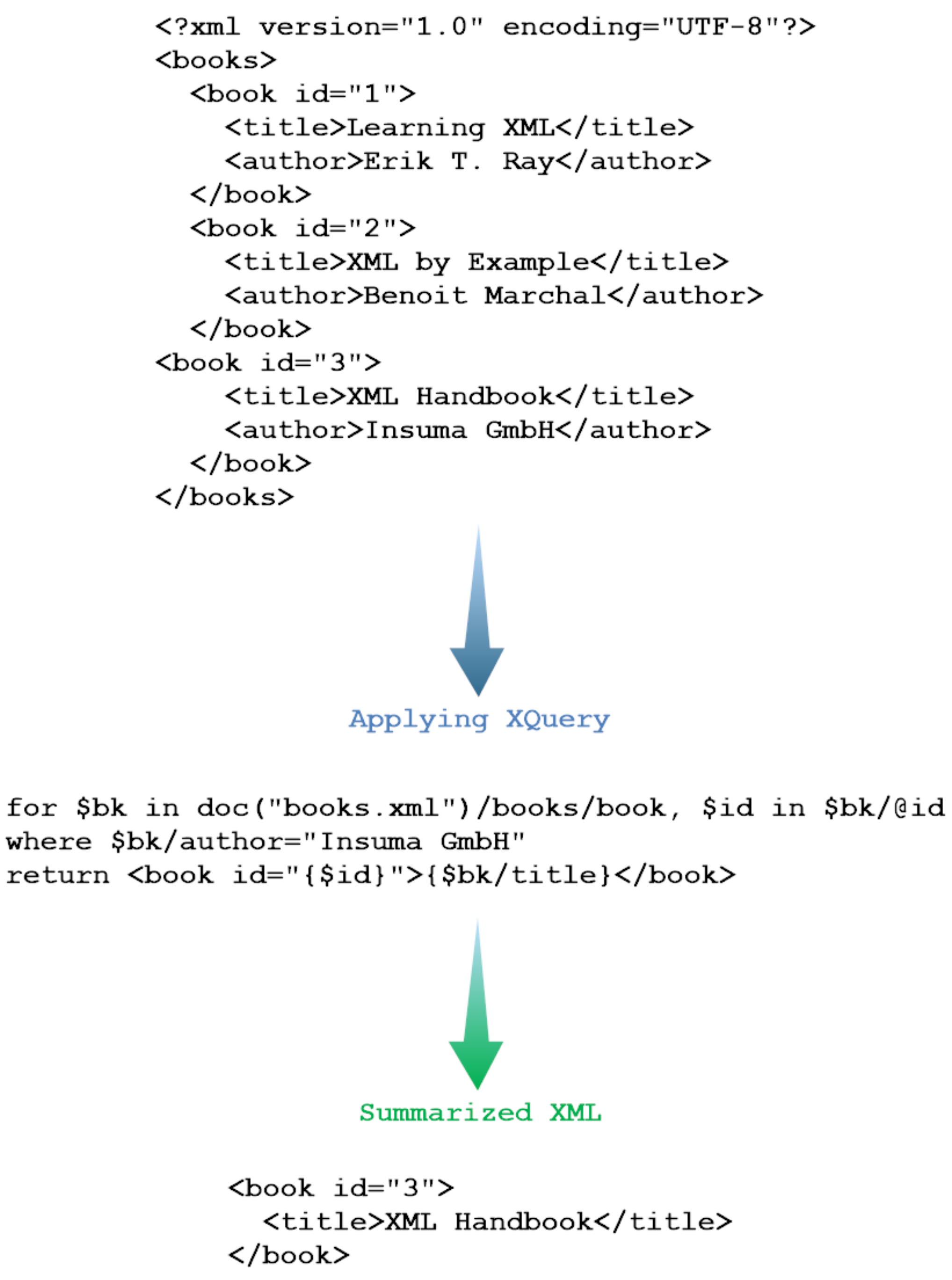


Figure 3.4. Approach of querying XDBMS.



```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book id="1">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
  <book id="2">
    <title>XML by Example</title>
    <author>Benoit Marchal</author>
  </book>
<book id="3">
    <title>XML Handbook</title>
    <author>Insuma GmbH</author>
  </book>
</books>
```

```
for $bk in doc("books.xml")/books/book, $id in $bk/@id
where $bk/author="Insuma GmbH"
return <book id="{$id}">{$bk/title}</book>
```

```
<book id="3">
  <title>XML Handbook</title>
</book>
```

Figure 3.5. Example of query summarization approach.

3.3. Challenges in XML Summarization.

A summary is helpful, which used to help users and applications to decide if a certain document is deserve to look into its fully content or not deserve. The good summary would summarize the most or all important information of the document as well as in many cases the summarized document used instead of the original one.

Furthermore, to generate such good summary, two goals must satisfy, which are: *maximum coverage* and *minimum space* as well as, the balance between the size of summarized document and its coverage is needed. However, to succeed in achievement to these goals, the following points must take in account [36]:

1. *Informativeness*. Any user will worry about informative information, such as elements and text. So, the important information in the document must presented concisely. This satisfy the goal of summarization process, which present salient points of the original document in concise abstract to the user.
2. *Non- redundant*. Redundancy in XML data and its structure are unmistakable. In listing 3.1 the same element `<keyword>` has occurred many times (once for keyword linked with a given data). So, instead of repeating all occurrences of elements in the summarized document, just represent them as a single element within all corresponding data as listed in listing 3.2.
3. *Coverage.* Coverage is exceedingly linked to the informativeness and denotes to the amount of information in the summarized document instead of the original data. Although, exclude certain elements from the original document, cause they aren't important enough. So, this excluding will improve the summarized document as well as its readability, but in contrast, the coverage is reduced.
4. *Coherence and Consistency.* The context or order of some elements sometimes need a logic, from where their parent and/or sibling element is important, such as in listing 3.1, the context of movies cannot have a production year element `<prod year>` without also having the title element `<title>` as well, but in contrast, movies can have the `<title>` without the `<prod year>`.

```
<movie>
  <title>Train:An Immigrant Journey</title>
  <aka>Train from Main Street</aka>
  <prod year>2000</prod year>
  <prod countries>
    <country>South Korea</country>
    <country>USA</country>
  </prod countries>
  <prod lang>English</prod lang>
  <prod location>New York City</prod location>
  <prod location>Queens</prod location>
  <director>Park, Hye Jung</director>
  <genres>
    <genre>Short</genre>
    <genre>Documentary</genre>
  </genres>
  <keywords>
    <keyword>Korean</keyword>
    <keyword>Pakistani</keyword>
    <keyword>Subway</keyword>
    <keyword>Manhattan</keyword>
 </keywords>
 <colourinfo>Color</colourinfo>
</movie>
```

Listing 3.1. XML Document of a movie [36].

```
<movie>
  <title>Train: An Immigrant Journey</title>
  <prod year>2000</prod year>
  <prod countries>
    <country>{South Korea, USA}</country>
  </prod countries>
  <prod location>{New York City, Queens}</prod location>
  <director>Park, Hye Jung</director>
  <genres>{Short, Documentary}</genres>
    <keywords>
     <keyword>{Korean, Pakistani, Subway,
               Manhattan}</keyword>
    </keywords>
</movie>
```

Listing 3.2. Concise Summary of listing 3.1 [36].

4. SIMILARITY MEASURES

String-based similarity measures are devided to character-based and string-based. The character-based such as N-gram, Smith-waterman, and Jaro; the string-based such as Dice's coefficient, Jaccard coefficient, and Cosine similarity (figure 4.1).

This thesis will focus on string-based (Dice's coefficient, the Jaccard coefficient and Cosine similarity), in contrast, character-based is out of scope.

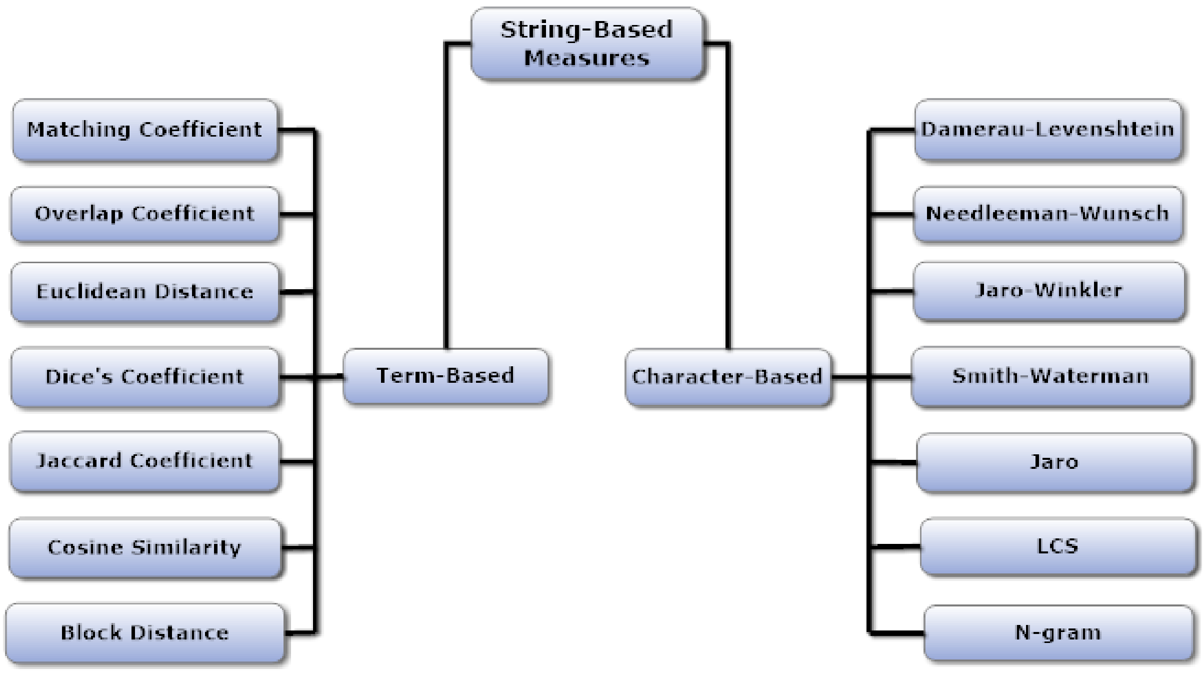


Figure 4.1. String-based measures.

Similarity measures have used mathematical methods to measure the similarity between two documents or query and collectoion of documents based on terms, usually such measures estimated regarding to their syntactical representation and distance metrics. The similar documents have large values and either zero or a negative values for dissimilar documents.

There are many methods for measuring text similarity according to query and document terms including Dice's coefficient, the Jaccard coefficient and Cosine similarity.

4.1. Dice's Coefficient

Dice's coefficient measures used to measure the similarity between two sets or two samples. Further, Dice's coefficient can be used to measure the similarity between two strings in terms of number of common bigrams. The bigram is a pair of neighboring letter in the string.

Dice's coefficient, defined in [37], is a statistical method to measure the similarity between queries and documents in terms of common n-grams. An n-gram is an adjacent section of letters in the string. Dice's coefficient is given in equation (1). The similarity values vary between 0 and 1.

$$\mathrm{Dice}(\mathrm{Q},\mathrm{D}) = \frac{2 \times \mathrm{n{-}grams}(Q) \cap \mathrm{n{-}grams}(D)}{\mathrm{n{-}grams}(Q) + \mathrm{n{-}grams}(D)} \qquad (1)$$

where *n-grams(Q)* are a multi-set of letter n-grams in query and *n-grams(D)* is a multi-set of letter n-grams in document.

Furthermore, the main idea is breaking a string to n-grams. For example, the string “right”, the set of bigrams would be {“ri”, “ig”, “gh”, “ht”}. Likewise, the string “write” would break down into {“wr”, “ri”, “it”, “te”}.

However, after bigrams have been created, the equation (1) can be applied. So, the set Q = {“ri”, “ig”, “gh”, “ht”}, then | Q | = 4 and D = {“wr”, “ri”, “it”, “te”}, then | D | = 4. The intersection of the bigram sets (Q ∩ D) is {“ri”}, only one element exists in the set. The union of the bigram sets (Q ∪ D) is {“ri”, “ig”, “gh”, “ht”, “wr”, “it”, “te”}, only seven elements exit in the set. So, according to equation (1), the similarity measurement between “right” and “write” is $\frac{2}{7}$.

4.2. Jaccard Similarity Coefficient

Jaccard coefficient is a static method used for comparing the similarity between sample sets. Jaccard similarity measure is the size of the intersection divided by the size of the union of the sample sets.

The Jaccard coefficient, described in [38], is a similarity measure between two sets, query and document, considered as a set of terms. The Jaccard similarity coefficient is calculated as in equation (2) by the intersection and union of both sets. The similarity values vary between 0 and 1.

$$J(Q, D) = \frac{|Q \cap D|}{|Q \cup D|} \qquad (2)$$

where the first set is query (Q) and the second set is document (D).

Furthermore, the most effective way of representation any documents as sets to identify the similarity between each other is to shingle the documents to sets of sentences or phrases that have common elements. No matter how sentences or phrases are ordered. A document consists of characters which have performed strings.

However, to make substrings of length *k* of any documents, *k*-shingle must defined. So, the set of *k*-shingles of documents can be associated to each other to determine the similarity. The value of *k* can be any constant, depends on the size of documents. Small documents such emails, *k* = 5, and large documents, such as research articles, *k* = 9 [39]. For example, following are query and document:

Q : *I am Hesham*
D : *Hesham I am*

Consider *k* = 2, because the document has small size, so the *k*-shingles for query and document are:

Q : {*I am*} {*am Hesham*}
D : {*Hesham I*} {*I am*}

According to equation (2), the Jaccard similarity measure is $\frac{1}{3}$.

4.3. Cosine Similarity

Cosine similarity measures the similarity between documents/collection and query. A collection of XML documents can be represented by a VSM in which each document is denoted by a vector of terms and their weights. A query, which's usually an expression that requests information from database, is represented as terms with weights to represent the importance of query terms. TF and IDF are used for the weighting of terms as statistical measures [40][41][42]:

1. The frequency of a term *j* in a document *i* ($tf_{i,j}$)
2. The frequency of the term *j* in the whole collection (df_j)
3. The inverse document frequency of term *j* in document *i* (idf_j)

Equation (3) gives the inverse document frequency of term *j* in the collection and equation (4) gives the weight of the term *j* in the document *i*.

$$idf_j = \log_e\left(1 + \frac{N}{df_j}\right) \qquad (3)$$

$$w_{i,j} = tf_{i,j} \times idf_j \qquad (4)$$

where *N* denotes the number of documents in the collection. Term weights using TF × IDF for measuring the similarity between query and document.

Cosine similarity is used to calculate the angle between query and document. If a vector is considered in a *V*-dimensional Euclidean space, the angle between query and document represents their mutual similarity. A smaller angle means greater similarity. Equation (5) defined the similarity between a document D_i and a query *Q*.

$$sim(Q, D_i) = \frac{\sum_{j=1}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{\sum_{j=1}^{V} w_{Q,j}^2 \times \sum_{j=1}^{V} w_{i,j}^2}} \qquad (5)$$

where $w_{Q,j}$ is weight of query term *j*, and $w_{i,j}$ is weight of term *j* in document *i* as mentioned in equation (4).

Consider the following example of a collection within three documents:

- D_1: *The New York Times.*
- D_2: *New York Post.*
- D_3: *Los Angeles Times.*

Table 4.1 shows that the collection of documents have transformed into VSM (calculations of TF, IDFand W according to equationa (3) and equation (4)).

Table 4.1. TF, IDF and W calaculations for documents.

	Term	*The*	*New*	*York*	*Times*	*Post*	*Los*	*Angeles*
Document (D_1)	***TF***	1	1	1	1	0	0	0
	IDF	0.602	0.398	0.398	0.398	0	0	0
	$\boldsymbol{W_{i,j}}$	0.602	0.398	0.398	0.398	0	0	0
Document (D_2)	**Term**	*The*	*New*	*York*	*Times*	*Post*	*Los*	*Angeles*
	TF	0	1	1	0	1	0	0
	IDF	0	0.398	0.398	0	0.602	0	0
	$\boldsymbol{W_{i,j}}$	0	0.398	0.398	0	0.602	0	0
Document (D_3)	**Term**	*The*	*New*	*York*	*Times*	*Post*	*Los*	*Angeles*
	TF	0	0	0	1	0	1	1
	IDF	0	0	0	0.398	0	0.602	0.602
	$\boldsymbol{W_{i,j}}$	0	0	0	0.398	0	0.602	0.602

Furthermore, to find the cosine similarity between a query and the collection of documents, which represented in table 4.1. Let's consider the following query (Q), which consists of two terms: *New Times*. Table 4.2 shows that the query has transformed into VSM (calculations of TF, IDFand W according to equationa 3 and equation 4).

Table 4.2. TF, IDF and W calculations for query.

	Term	*The*	*New*	*York*	*Times*	*Post*	*Los*	*Angeles*
Query (Q)	***TF***	0	1	0	1	0	0	0
	IDF	0	0.301	0	0.301	0	0	0
	$\boldsymbol{W_Q}$	0	0.301	0	0.301	0	0	0

Finally, table 4.3 shows the calculated cosine similarity according to equation (5) and figure 4.2 shows the vector space model. The document D_1 is the best fit to the query and the ranking is D_1, D_2, D_3.

Table 4.3. Cosine calculations.

Cosine Similarity	
Cosine(Q,D_1)	1
Cosine(Q,D_2)	0.707
Cosine(Q,D_3)	0.707

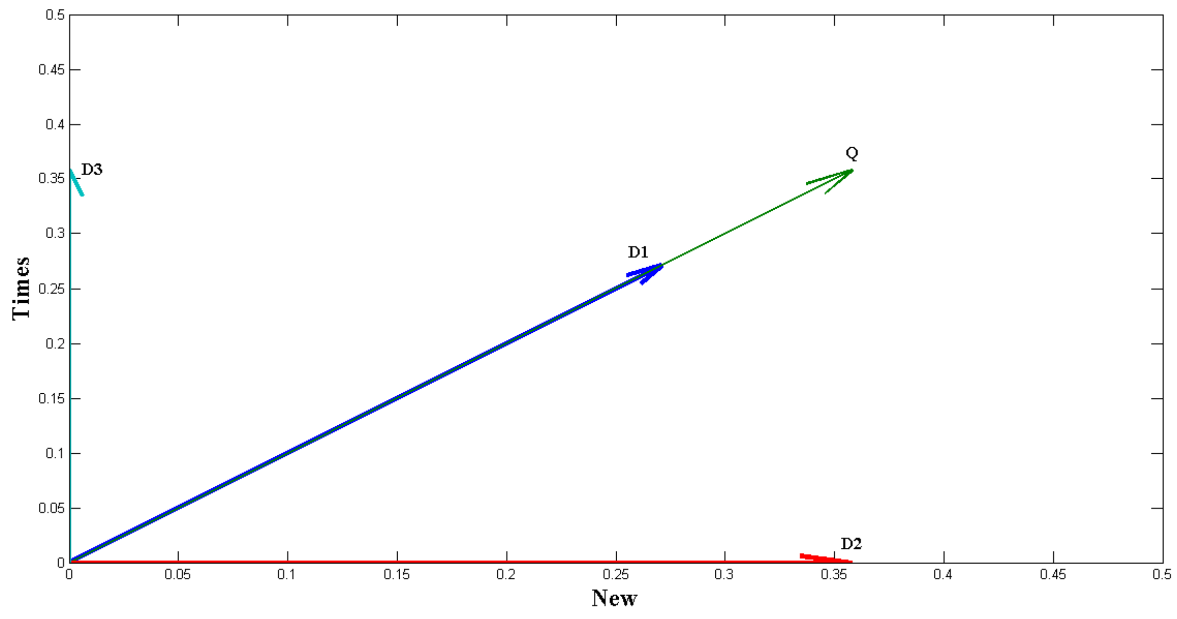


Figure 4.2. Vector space model.

5. RELATED WORKS

There is currently a high level of interest within the research community for text processing of Arabic documents as well as queries, stemmers, ranking, keyword extraction, XML document summarization and securing.

Text summarization process uses extraction and abstraction methods, and performs the analyzing of a source text, determining its important information (salient points), and gathering an appropriate text summary. Text summarization can be presented as indicative, informative, or critical. In a single document summarization, a ranking framework is functioning much better than a classification framework. Furthermore, text summarization process focused on plain text in text datasets, which is not always fit and applicable to XML summarization process, because of XML structure and semantic information [43][44].

XML schema summarization is concerning about summarizing XML schemas instead of XML documents. A schema summary process can provide a precise abstract of the entire schema including salience points with no redundant elements, and gives a possibility to explore only the relevant schema components [45].

XML structure summarization is another related topic which summarizes XML structure using clustering rather than summarizes XML document. After distance metrics clustering, this method modeled any XML document as hierarchal rooted tree to address clustered groups that have similar structure [46].

The size of XML documents is the most concerned disadvantage. This will cause increasing in storage space, increasing in transmission bandwidth, and increasing in processing time and memory, such as parsing and updating. So, the inevitable compression algorithms, such as Relax NG, XMl, gzip, bzip2 and Huffman are introduced to reduce the document size [47][48].

Another citations focused on statistical summarization based on query estimation. So, the statistics are obtained from exploration of schema transformation and schema validation of an XML document that match the estimated query, such as StatiX XML schema in [49], which’s used histograms statistical method to summarize structure and content of the XML document.

In exploration process of complex and large XML data on-line using XML query languages, such as XQuery and XSLT in relational database systems. Effective methods needed to reduce query processing time. So, an approximate query answers mechanism has been used to reduce the processing time and provide feedback to the end user. TreeSketch mechanism in [50] is a summarization mechanism that used for this purpose. Based on [50] TreeSketch provides a concise and fast approximate tree-structured answers of queries.

Wei Wang and et al. presented a framework of an optimization method to estimate the XML path selectivity expressions in dynamic context. They proposed XML data structure over a bloom histogram to approximate the distribution of XML path frequency and estimated the XML path selectivity [51].

Ning Zhang and et al. proposed XSEED synopsis to solve the problem of cardinality estimation for XML path queries in relational database systems. The synopsis is a compact graph structure resulted from summarization of an XML document. XSEED focuses on structural of XML to estimate the cardinality for path queries that only contain structural constraints using a graph-based algorithm [52].

Mayorga V. and Polyzotis N. addressed the problem of approximation answer of XML query over data of XML stream, such as RSS feeds. They proposed sketch-based XML stream synopsis technique, which summarizes XML data streams instead of summarizes XML documents [53].

A template-based is a semi-automatic approach that used to summarize XML documents. The end user can design a proper template that matches semantics of the XML documents and defines the rules for summarization process, or the template can be generated automatically. This template is used in automatic summarization process. The template-based approach is presented in [54].

M. Ramanath and K. Kumar are covered the generation of XML summaries according to generic summaries of XML documents based on their contents. They proposed an automatic technique to generate concise readable summaries based on memory budget. The summary document contains the important information of the original document according to elements importance; coherent and concise representation of important elements; and generating a semantic summary that suits different memory budgets [36].

EXsum is presented by *Jose de Aguiar and Theo Harder*. EXsum is a framework used to summarize properties of an XML document based on statistical information of XPath axes related to a specific element in the document [55].

XML documents, databases, querying XML data using XQuery and XML indexing (which summarize large XML data structure into a tree) are discussed in [56] and [57].

The retrieval of formal Arabic language, as used in media such as news domains, as well as the retrieval of Arabic dialects is among the problems that face information retrieval systems. Natural language processing of Arabic information to enable retrieval is considered in [21].

Different Arabic text stemmers, as well as constructed Arabic stopwords lists used in information retrieval systems, are described in [23] and [25].

Stemming methodologies and query terms affect the information retrieval systems according to the word and stem. In contrast, term importance can be computed according to term frequency and inverse document frequency as described in [40].

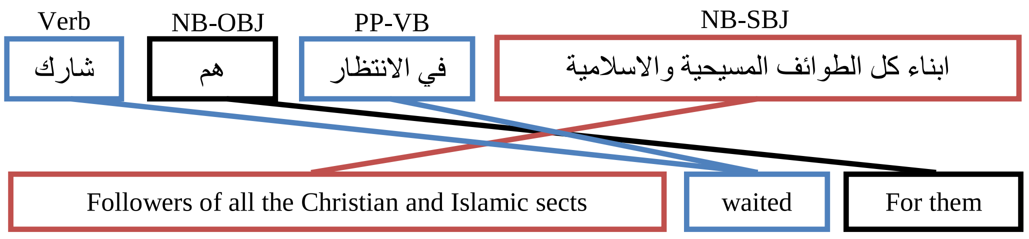
The use of similarity measures in a vector space model, according to TF and IDF of documents and structural of terms, is described in [41].

Automatic keyword extraction according to candidate keywords (that are extracted from a document and selected based on TF of words within these documents), word degree and ratio of degree to frequency are covered in [58].

Arabic natural language processing techniques have used linguistic resources such as Corpora and Lexicon to develop parser and POS-tagger. This has enabled the creation and evaluation of a framework for use in Islamic sciences written in the Arabic language. This framework could adapt the theories, resources, tools and applications of other NLPs such as English and French as described in [59].

Three vector space models (Cosine, Dice and Jaccard coefficients) for classifying Arabic text using the K-Nearest algorithm and the IDF term are compared in [60].

Currently, there is a high level of activity in the production of tools that provide automatic annotation and translation of Arabic texts. The linguistic difference of the Arabic language to western culture languages results in complexity of implementation. In [NP Subject Detection in Verb-Initial Arabic Clauses] the focus is on the words-in-sentence ordering problem and the different way Arabic phrases are formed. For example the sentence in figure 5.1, which in English is ordered from left-to-right compared to the Arabic phrase which is ordered right-to-left, illustrates the ordering problem [61].



Verb
NB-OBJ
PP-VB
NB-SBJ
شارك
هم
في الانتظار
ابناء كل الطوائف المسيحية والاسلامية
Followers of all the Christian and Islamic sects
waited
For them

Figure 5.1. Phrase reordering [61].

Barbara Carminati and et al. focused on confidentiality and completeness to secure XML data. They proposed a comprehensive framework for secure outsourcing of XML data using different techniques of digital signature and encryption strategies [62].

Node filtering model used to evaluate user queries by accessing the security policies of XML datasets. Query rewriting system model transformed user queries using authorization rules and evaluated the transformed queries over the original XML datasets. Both models were combined and described in [63].

Data exchange over the Internet/network needs to be secure, authentic, private and confident. So, *Seifedine Kadry* presented an approach to cipher information of a text document that needed in data exchange. A cipher algorithm used to encrypt the text document. The encrypted document sent as encrypted webpage, which's created using DOM parser [64].

Participant domain name token of web services security standards which's used to authenticate a message sender location. Participant domain name token, decryption, signature, XML and their relationships are presented in [65].

Secure and efficient query emphasis in external and distributed XML databases (data repositories) as well as time stamps and FLWOR expressions are presented in [66].

Security in XML data transmission, such as confidentiality and integrity and XML data encryption based on importance and sensitivity proposed in [67].

Securing XML documents using user identify certificates and keys (public key and private key cryptography) based on Java platform are described in [68].

P. Dhivya and et al. proposed CMWA routing protocol to find reliable route between source and destination based on mobility and energy aware metric in clustered Ad-hoc networks over SOA platform. They applied the XML data security techniques to encrypt the secure data over reliable route and the encrypted SOAP message used to exchange the data between the mobile nodes [69].

6. OBJECTIVE, HYPOTHESIS AND METHODOLOGY

Updating XML document causes extreme increasing in document size with time, this growth in its turn needs more storage memory and more powerful capability processors to process, there is a need to develop and build a reliable mechanism to access XML documents in local or remote computers and networks easily, such as development of XML schema summarization technology, XML structure summarization technology, XML statistics summarization technology and querying XML database systems approaches, which concern about ranking the resulted XML document as well as comparing similarity with query.

The research is going to explore the systematization of Arabic XML documents summarization knowledge and securing Arabic XML documents and their presentation to find some mechanisms to satisfy the needs of XML summarization and to protect the summarized document.

6.1. Research Objective

The research objective focused on improvements of existing summarizing and securing evaluations approaches to reflect the requirements of XML summarization process as well as querying, ranking and securing. However, the research objectives are:

1. Summarizing Arabic text and/or Arabic XML documents according to:

    - Character encoding.
    - Stemming process.
    - Document size.
    - Information content.
    - Information importance.
    - Security issues.

2. Ranking the summarized text and/or documents based on:

    - Queries.
    - Similarity measures.

3. Applying security issues on XML documents according to:

   - XML digital signature, which includes creation and verification processes.
   - XML document encryption and decryption process.

4. Design and build a system model, which will establish these objectives.

6.2. Research Hypothesis

Our general hypothesis is particularly in the domain of XML data representation, information retrieval systems, summarization of Arabic XML documents, XML queries, ranking Arabic XML documents and securing Arabic XML documents. However, the research hypotheses are:

1. Processing of Arabic text that contains Arabic words with different forms of syntax and morphologies as well as different meanings. Grammatically, documents contain different forms of words including derivations. This causes problems in text processing, document summarization and information retrieval systems.
2. There is a high level of information loss during the processes of querying, document summarizing and information retrieval, especially with large documents, as information loss is directly proportional to the size of documents during these processes.
3. Generating a good summarized document without losing important information.
4. Summarization process leads to minimize consumption of memory.
5. Ranking summarized XML documents are helpful in information retrieval systems.
6. The similarity measures between XML documents and their summarized documents are close.
7. The similarity between query and its result depending on the terms of query and content of summarized XML document.
8. Security issues are powerful.
9. XML digital signature and encryption affect the summarized XML document.

6.3. Research Methodology

Objectives and hypotheses of the research will lead us to develop a new methodology that will be able theoretically to evaluate various kinds of Arabic XML documents from different domains and different categories. Theoretical evaluation presents a powerful methodology to analyze the influence of XML structures and its contents over our model.

This research describes a model system, which's designed for ranking Arabic documents stored in the different formats in information retrieval processes and security model to protect user queries and documents.

The research results should help engineers, network administrators, database designers and information retrieval systems to deal with Arabic XML documents.

Furthermore, methodologies in [16 - 69] can guide this research to achieve the objectives.

7. RAX SYSTEM MODEL

As mentioned in section 6.2, the Arabic word has different forms of syntax and morphologies with different meanings. Grammatically, documents contain different forms of words including derivations. This causes problems in text processing, document summarization and information retrieval systems. Furthermore, there is a high level of information loss during the processes of querying, document summarizing and information retrieval, especially with large documents, as information loss is directly proportional to the size of documents during these processes.

This research proposes an RAX system (Ranking Arabic XML documents system) to solve the problem of Arabic text processing. The RAX system as shown in figure 7.1 is designed for ranking Arabic documents stored in the different formats and different categories based on user query. RAX system has two stages, which are a preparation stage and an implementation stage. RAX system stages are described in following sections with more details.

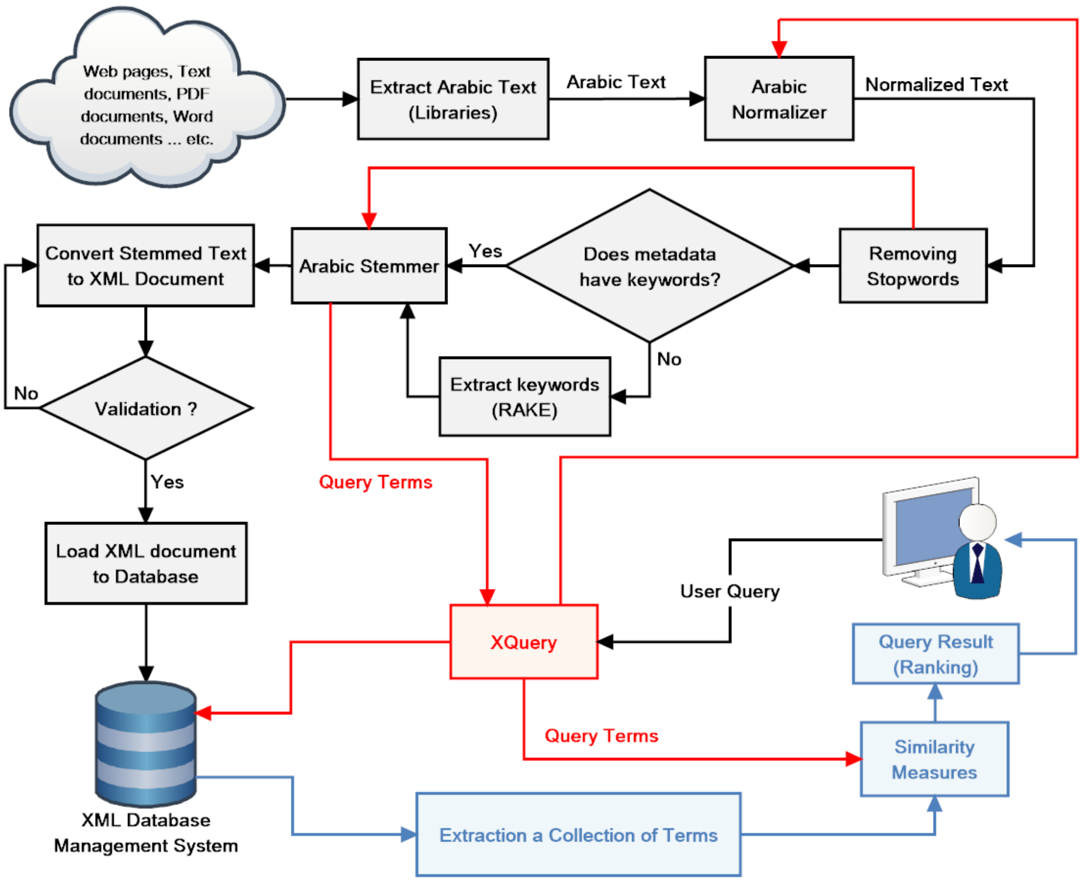

Web pages, Text documents, PDF documents, Word documents ... etc.
Extract Arabic Text (Libraries)
Arabic Text
Arabic Normalizer
Normalized Text
Removing Stopwords
Does metadata have keywords?
Yes
No
Arabic Stemmer
Extract keywords (RAKE)
Convert Stemmed Text to XML Document
Validation ?
No
Yes
Load XML document to Database
Query Terms
XQuery
User Query
Query Terms
Query Result (Ranking)
Similarity Measures
Extraction a Collection of Terms
XML Database Management System

Figure 7.1. RAX system model.

7.1. Preparation Stage

The preparation stage (see figure 7.2) begins with the loading of different Arabic webpages and Arabic documents such as word, PDF and text documents into a text extracting libraries. This process is described in the following seven steps:

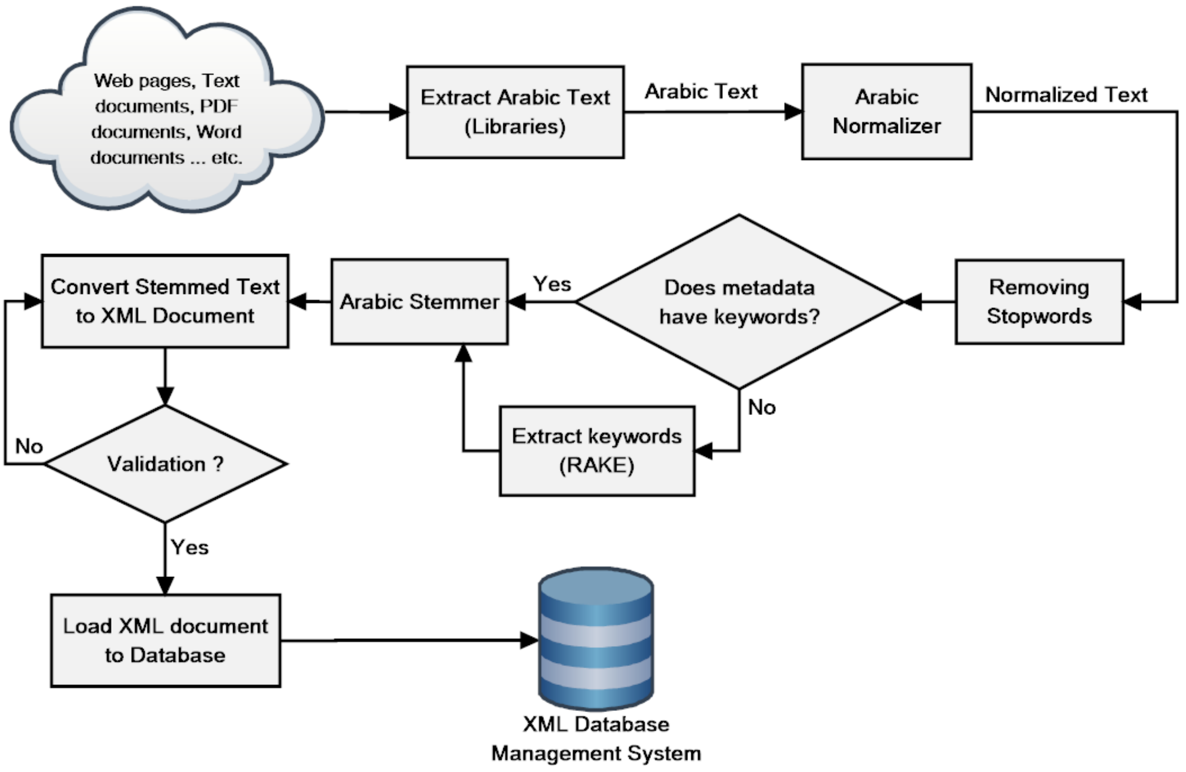

Web pages, Text documents, PDF documents, Word documents ... etc.
Extract Arabic Text (Libraries)
Arabic Text
Arabic Normalizer
Normalized Text
Removing Stopwords
Does metadata have keywords?
Yes
No
Arabic Stemmer
Extract keywords (RAKE)
Convert Stemmed Text to XML Document
Validation ?
No
Yes
Load XML document to Database
XML Database Management System

Figure 7.2. Preparation stage of RAX system.

1. The extracting libraries are written in Java and used in many advanced content management tools (e.g. Alfresco, Lucene, Apache Tika and REWOO Scope). The extracting libraries that have been used by RAX system to extract the text are:

   - jsoup Java HTML Parser, which is a Java library that used to extract text from web pages of HTML format [70].

   - Apache POI Word-Text Extraction [71] and Apache Tika [72] are used to extract text and metadata from word documents, respectively.

   - In simple text documents, the text extraction process done by reading the content of the documents line by line.

- Apache PDFBox [73] and iText [74] are the class libraries used to extract of pure text and metadata from PDF documents.

2. An Arabic normalizer performs the normalization process in which Arabic diacritics, punctuation, non-letters and stretching letters (kashidas) are removed and different versions of a letter are converted into the standard letter as we mentioned in section 2.2.1. For instance the letters أ , إ , آ , اً are all forms of the letter ا (letter *A* in the English language). These various forms would each be converted into letter ا. Other examples are the normalization of the letters ى and ه which are transformed into ي and ة. A diacritic word مَدْرَسَةٌ meaning *school* will normalize to مدرسة without diacritics. The stretched word كِتَـــــــــاب meaning *book* will normalize to كتاب without stretching. The word أحَمّدْ contains diacritics and one form of letter ا . This normalizes to احمد with the standard form of letter ا and without diacritics and so on.

3. The RAX system then removes Arabic stopwords, such as من، في، الى، يلي، ضد، بعد، ان، حتى، وهو، يكون به، وليس، أحد، على، وكان، تلك، كذلك، التي، وبين، فيها، عليها، وعلى، لكن، عن، مساء، منذ، الذي، أما، حين، ومن، لا، ليسب، وكانت، ما، عنه، حول، دون، مع، لكنه، ولكن، له، هذا، والتي، فقط، ثم، هذه، أنه، تكون، قد، بين، جدا، لن، نحو، كان، لهم، لأن، اليوم، لم، هؤلاء، فإن، فيه، ذلك، لو، عند، اللذين، كل، بد، لدى، وثي، أن، ومع، فقد، بل، هو، عنها، منه، بها، وفي، فهو، تحت، لها، أو، إذ، علي، عليه، كما، كيف، هنا، وقد، كانت، لذلك، أمام، هناك، قبل، معه، يوم، منها، إلى، إذا، هل، حيث، هي، اذا، او، و، مالا، الي، إلي، مازال، لازال، لايزال، ما يزال، اصبح، أصبح، أمسى، امسى، أضحى، اضحى، ظل، ما برح، ما فتئ، ما انفك، بات، صار، ليس، إن، كأن، ليت، لعل، لاسيما، ولايزال، الحالي، ضمن، اول، وله، ذات، أي، بدلا، اليها، انه، الذين، فانه، وان، والذي، وهذا، لهذا، الا، فكان، ستكون، مما، أبو، بإن، اليه، يمكن، بهذا، لدي، وأن، وهي، وأبو، آل، هن and الذى. A list of Arabic stopwords has been created including pronouns and prepositions according to[22].

4. Following normalization and removal of stopwords. If there are no keywords in the document's metadata the RAX system uses RAKE (rapid automatic keyword extraction) technology to extract keywords from text. RAKE contains a list of stopwords, phrase and word delimiters that is used to identify candidate keywords – a series of words by priority of occurrence in the text. Each candidate keyword is scored according to the ratio of word degree to word frequency. The top scoring candidates are selected as keywords, which calculated as $\frac{1}{3}$ number of words [58].

Since it is difficult to process Arabic language in summarization and information retrieval due to its complex morphology, an Arabic stemmer is used to reduce derivational forms of a word to a stem or a root word (base form) as mentioned in section 2.2.1. Each root gives rise to many different words, such as nouns, adjectives, and verb stems. For example the words مَكْتَبْ (maktab) *office*, كُتُبْ (kutub) *books*, كِتَابْ (kitAb) *book*, كَتَبَ (kataba) *he wrote*, and نَكْتُبُ (naktubu) *we write* all come from the root كتب (ktb). The *RAX* system uses its own stemmer to strip off prefixes such as لل، ل، ا، و، س، ب، ي، ن، م، ت and ف and suffixes such as هما، تما، كما، ان، ها، وا، تم، كم، تن، كن، نا، تا، ما، ون، ين، هن، هم، ته، تي، ني، ن، ك، ه، ة، ت، ا، ي and ات. To illustrate the processes mentioned above, fragments of three documents and their translations in table 7.1 are used:

Table 7.1. Fragments of three documents.

Document	Content	Translation
D_1	الأنظمة الحكومية هي الأنظمة الموثوق بها والتي تحتفظ بمعلومات عن المواطنين ...	*Government systems are the trusted systems that hold information about citizens….*
D_2	قواعد البيانات هي الجوهر لنظم المعلومات ...	*Data bases are the core of information systems. ….*
D_3	قواعد البيانات في أنظمة الحكومة التي تحتفظ بمعلومات عن المواطنين وهم جوهر انظمة المعلومات ...	*Data bases in government systems hold information about citizens and they are the core of these information systems. ….*

In the first step the system eliminates stopwords and the sentences are modified as follows in table 7.2:

Table 7.2. Eliminate stopwords from fragments of three documents.

Document	Content	Translation
D_1	الأنظمة الحكومية الأنظمة الموثوق تحتفظ بمعلومات المواطنين	*Government systems trusted systems hold information citizens*
D_2	قواعد البيانات الجوهر لنظم المعلومات	*Data bases core information systems*
D_3	قواعد البيانات أنظمة الحكومة تحتفظ بمعلومات المواطنين جوهر انظمة المعلومات	*Data bases government systems hold information citizens core information systems*

In the next step (stemming) all of the words are transformed into normal form. In this way the sentences are put into their final form as follows in table 7.3:

Table 7.3. Stemming fragments of three documents.

Document	Content	Translation
D$_1$	نظم حكم نظم وثق حفظ علم وطن	*Government system trust system hold information citizen*
D$_2$	قعد بين جوهر نظم علم	*Data base core information system*
D$_3$	قعد بين نظم حكم حفظ علم وطن جوهر نظم علم	*Data base government system hold information citizen core information system*

5. Document Creation; XML is a simple textual data, which supports different Unicode standards for different languages and well as benefiting from simplicity and usability over the Internet. XML syntax is widely used as a default format to represent data structure and create documents e.g. in Microsoft Office, OpenOffice.org, and web services. By this stage RAX has initialized the XML document and converted normalized stemmed Arabic text originating from PDFs to well-formed Arabic XML documents using Java API for XML Processing [75] and Simple API for XML [76]. Listing 7.1 shows the fragment of the first document (D$_1$) in this step represented in XML form.

6. The XML database management systems enable storage of XML documents and transfer of data between relational databases. These documents can be queried, transformed, transported and returned to a calling system. So after the documents are preprocessed and XML formed they are ready to be stored in an XML DBMS. The Sedna DBMS, which has full ability of database services and gives flexible XML processing facilities including W3C XQuery accomplishment with full-text search, is used. This is the end of preparation stage and *RAX* system is ready for implementation.

```
<?xml version="1.0" encoding="UTF-8"?>
<articles>
   <article id="1">
      <title>الانظمة الحكومية</title>
      <author>محمد نصر</author>
      <subject>سياسة</subject>
      <keywords>نظام الحكم</keywords>
      <statistics>
         <percentstemmed>100.0</percentstemmed>
         <stemingtime>0.30</stemingtime>
         <stemmedwords>7</stemmedwords>
         <nonstemmedwords>0</nonstemmedwords>
         <stopwords>4</stopwords>
         <punctuationwords>0</punctuationwords>
         <nonletterwords>0</nonletterwords>
         <totalwords>11</totalwords>
         <totalstemmedwords>7</totalstemmedwords>
      </statistics>
      <stemmedtext>وطن حفظ علم نظم حكم نظم وثق </stemmedtext>
      <notstemmedwords/>
   </article>
</articles>
```

Listing 7.1. XML form of fragment D_1.

7.2. Implementation Stage

One of the obvious facts about information retrieval systems, as opposed to sorting and searching algorithms, is that the more documents are stored into the database the better it performs. Regarding the system complexity and hardware limitations the collection of 100 documents was found to be optimal for the different scenarios used in the research. Next is a description how the system works during implementation (see figure 7.3):

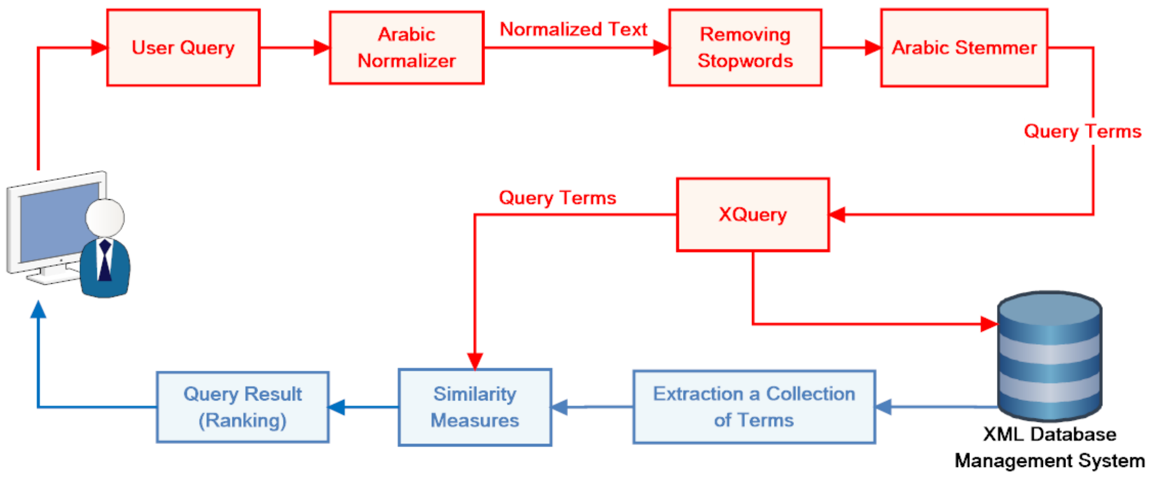


Figure 7.3. Implementation stage of RAX system.

1. When the end user enters a query the *RAX* system performs its processing in the same way as with documents (normalization, the removing of stopwords and stemming). As a result the query expression is transformed in to vector of terms.

2. Next the *RAX* system executes XQuery on the document base in the XML DBMS (Sedna DBMS) expression which includes the vector of query terms. XQuery uses XPath syntax for accessing different nodes of XML documents. A set of XML documents is returned as a result of the query. Listing 7.2 shows an XQuery expression; this query returns a collection of documents including the term frequency for each document that contains the query term.

```
let $query_term :='نظم'
for $document_terms in //article, $id in $ document_terms/@id
let $output_terms := tokenize($document_terms/stemmedtext/text(), $query_term)
let $freq := count($output_terms)where $freq>0
return <document id="{$id}">{<frequency>{$freq}</frequency>}</ document>
```

Listing 7.2. XQuery expression used by *RAX* system.

3. To rank documents the *RAX* system calculates weights for each particular term in the document. Term frequency and inverse document frequency are used for this purpose (equation (3) and equation (4) in section 4.3). This means that the vector space model in which the documents are transformed is enriched with additional information: each document's vector is represented by an array of term-weight pairs. The user query is processed in the same way. Final comparison between these two is performed using cosine similarity as a measure of the documents' ranking (equation (5) in section 4.3).

The following example shows how the documents' samples fragments (described in section 7.1- stage 5) are used in this process. Transformation in the improved vector model is the most crucial and processor intensive phase. After this each fragment of document being considered for ranking is represented with two vectors. The original XML document consists of a vector of terms. The terms are collections of words and each word is represented by its weight (TF*IDF) value. In this way the documents are represented with two vectors. Original words are filtered and transformed into normal form and for convenience are labeled terms. For clarity, TF and IDF are represented separately i.e. (term_1, tf_1, idf_1), …, (term_n, tf_n, idf_n), where n is the full number of terms in the document set. Thus there are 9 different terms for our example and the VSM for each document should contain these. TF is represented by row frequency, which represents the number of occurrences of a specific term in the document, and IDF is calculated according to equation (3) in section 4.3. See table 7.4.

Table 7.4. TF and IDF calculations for samples used in example.

Document: D_1									
Term	نظم *System*	حكم *Government*	وثق *Trust*	حفظ *Hold*	علم *Information*	وطن *Citizen*	بين *Data*	قعد *Base*	جوهر *Core*
TF	2	1	1	1	1	1	0	0	0
IDF	0.30102	0.39794	0.60206	0.39794	0.30102	0.39794	0	0	0
W_{D1}	0.60204	0.39794	0.60206	0.39794	0.30102	0.39794	0	0	0
Document: D_2									
Term	نظم *System*	حكم *Government*	وثق *Trust*	حفظ *Hold*	علم *Information*	وطن *Citizen*	بين *Data*	قعد *Base*	جوهر *Core*
TF	1	0	0	0	1	0	1	1	1
IDF	0.30102	0	0	0	0.30102	0	0.39794	0.39794	0.39794
W_{D2}	0.30102	0	0	0	0.30102	0	0.39794	0.39794	0.39794
Document: D_3									
Term	نظم *System*	حكم *Government*	وثق *Trust*	حفظ *Hold*	علم *Information*	وطن *Citizen*	بين *Data*	قعد *Base*	جوهر *Core*
TF	2	1	0	1	2	1	1	1	1
IDF	0.30102	0.39794	0	0.39794	0.30102	0.39794	0.39794	0.39794	0.39794
W_{D3}	0.60204	0.39794	0	0.39794	0.60204	0.39794	0.39794	0.39794	0.39794

The next step is to determine the cosine similarity between the query and the previous collection which is represented in table 7.4. Let us consider a query which contains two words: **نظم علم - *information system*** (in stemming form). Table 7.5 shows that the query has transformed into VSM.

Table 7.5. TF and IDF calculations for query.

Query (Q) نظم علم - Information System									
Term	نظم *System*	حكم *Government*	وثق *Trust*	حفظ *Hold*	علم *Information*	وطن *Citizen*	بين *Data*	قعد *Base*	جوهر *Core*
TF	1	0	0	0	1	0	0	0	0
IDF	0.30102	0	0	0	0.30102	0	0	0	0
W_{Q1}	0.30102	0	0	0	0.30102	0	0	0	0

Finally, table 7.6 shows the calculated cosine similarity according to equation (5) in section 4.3. The document D_3 is the best fit to the query and the ranking is D_3, D_1, D_2.

Table 7.6. Cosine calculations.

Cosine Similarity	
Cosine(Q, D_1)	0.52230
Cosine(Q, D_2)	0.45894
Cosine(Q, D_3)	0.64904

7.3. Practical Evaluation of the RAX System

As previously mentioned in implantation stage section, the experiment was carried out on a collection of 100 of random documents from different categories (https://en.wikipedia.org/wiki/Portal:Contents/Categories). Detailed results are given in appendix. The next table (table 7.7) illustrates these categories:

Table 7.7. Categories of interest.

Category	**No. of documents**
General reference	1
Culture and the arts	2
Geography and places	1
Health and fitness	3
Mathematics and logic	5
Natural and physical sciences	9
People and self	11
Philosophy and thinking	3
Society and social sciences	21
Technology and applied sciences	44
Total	**100**

Four queries from different domains (Computer Science, Ecology and Social Science) are used in the experiment in order to cover all of the documents. The *RAX system* is used for measuring similarity by using TF, IDF and cosine similarity as previously described. Following are the queries and results:

1. Query$_1$= {الشبكات السلكية واللاسلكية - *Wire and wireless networks*}, after stemming process has taken place, Query$_1$={شبك سلك سلك - *Wire Wire Network*}. In appendix table 1, query$_1$ has two terms; the (*Wire*) term which occurred twice (TF=2) and the (*Network*) term which occurred once (TF=1). So the inverse document frequency and the weight of query$_1$ were determined from equations (3) and (4) in section 4.3. As a result of query$_1$ in appendix table 2, we have found 60 documents in total. Eighteen documents contain both terms (*Wire* and *Network*); 9 individual documents contain the term (*Wire*), and 33 individual documents contain the term (*Network*). The cosine similarity measures are calculated between query$_1$ terms and document terms according to equation (5) in section 4.3. From appendix table 2, we conclude that 60% of the collection matches query$_1$. The total of terms' frequencies and the weights for both terms are equal to the total in appendix table 9. The top ranked documents which contain both terms are D$_{35}$, D$_{63}$, D$_{58}$, D$_{20}$, D$_{50}$, D$_{54}$ and D$_{11}$ (see figure 7.4).

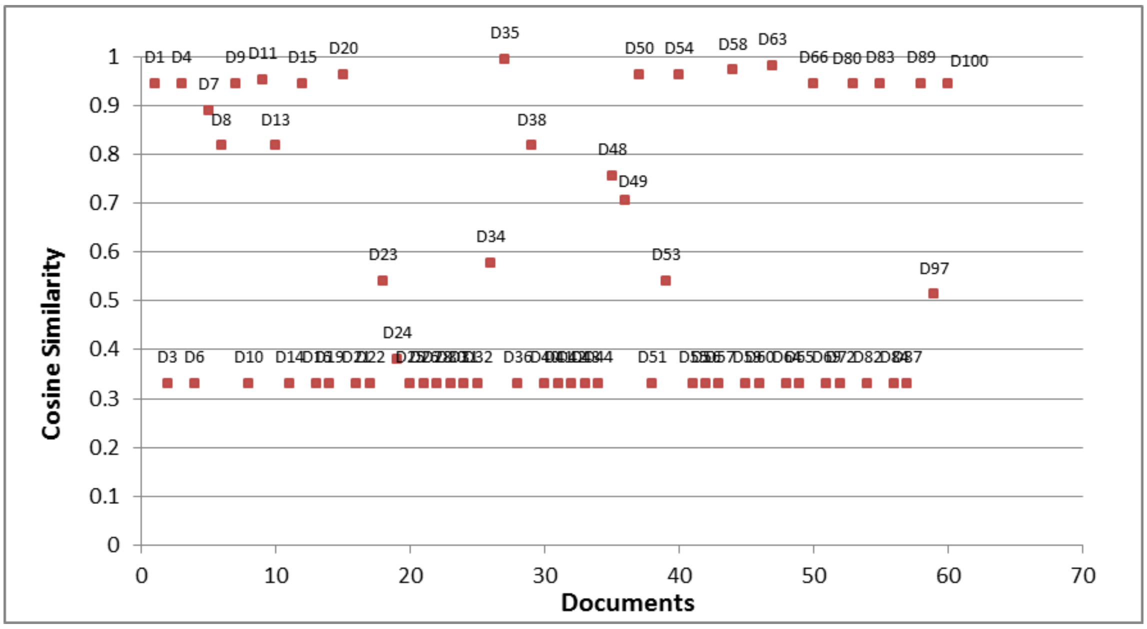


Figure 7.4. Similarity measures of query$_1$.

2. Query$_2$= {صيانة وبرمجة الكمبيوتر - *Maintenance and Computer Programming*}, after stemming process has taken place, Query$_2$={صون برمج كمبيوتر - *Maintain Computer Program*}. In appendix table 3, query$_2$ has three terms; the (*Maintain*) term which occurred once (TF=1), the (*Computer*) term which occurred once (TF=1) and the (*Program*) term which also occurred once (TF=1). So the inverse document frequency and the weight of query$_2$ were determined from equation (3) and (4) in section 4.3. As a result of query$_2$ in appendix table 4, we have found 47 documents in total. Ten documents contain both terms (*Maintain* and *Program*); 22 individual documents contain the term (*Maintain*), and 15 individual documents contain the term (*Program*). As none of the collection match the term (*Computer*) it is impossible to calculate IDF due to the denominator df_j being zero so the *RAX* system has excluded the term (*Computer*) from further consideration. The cosine similarity measures are calculated between query$_2$ terms and document terms according to equation (5) in section 4.3. We conclude that the *RAX* system will exclude terms which are not matched. From appendix table 4, we conclude that 47% of the collection matches query$_2$. The total of terms' frequencies and the weights for both terms are equal to the total in appendix table 9. The top ranked documents which contain both terms are D_{62}, D_{29}, D_{33}, D_{16}, D_{58}, D_{11}, D_{50}, D_{56}, D_{53} and D_{36} (see figure 7.5).

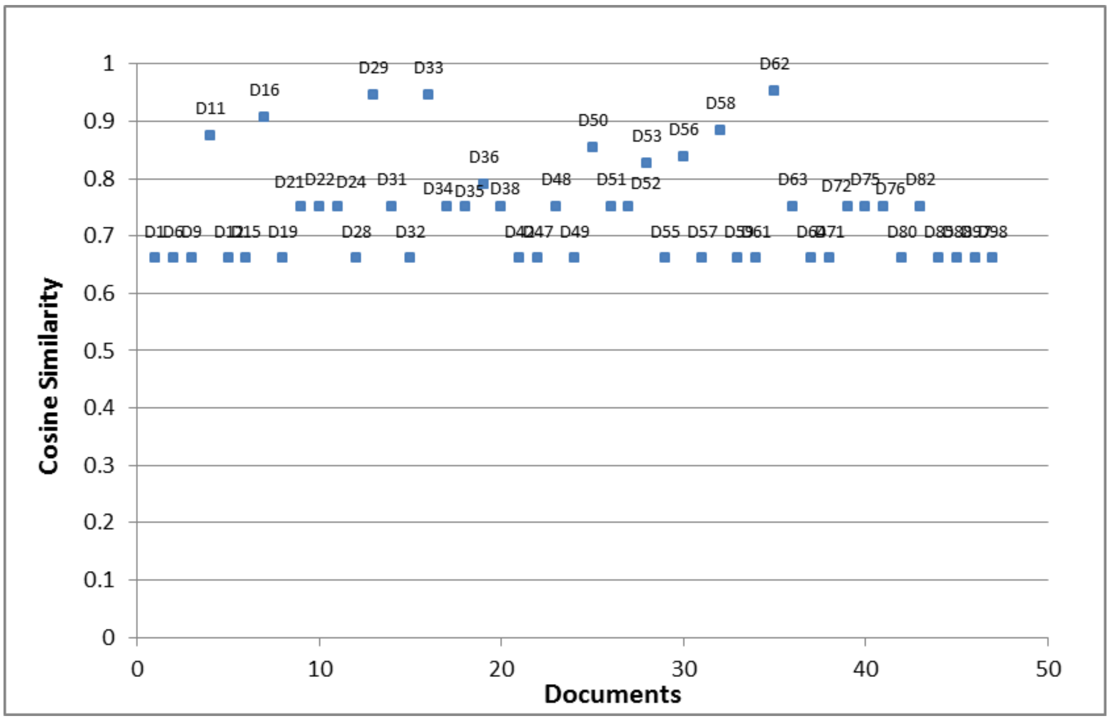


Figure 7.5. Similarity measures of query$_2$.

3. Query$_3$= {اللغة العربية والوطن العربي - *The Arab home and Arabic language*}, after stemming process has taken place the Query$_3$= {لغا عرب وطن عرب - *Arab home Arabic language*}. In appendix table 5, query$_3$ has 3 terms; the (*Language*) term occurred once (TF=1), the (*Arab*) term occurred twice (TF=2) and the (*Home*) term occurred once (TF=1). So the inverse document frequency and the weight of query$_3$ were determined according to equations (3) and (4) in section 4.3. As a result of query$_3$ in appendix table 6, we have found that 79 documents in total from the collection match query$_3$ terms. Twenty-nine documents contain terms (*Language*, *Arab* and *Home*); 5 documents contain the terms (*Language* and *Arab*); 7 documents contain the terms (*Language* and *Home*); 13 documents contain the terms (*Arab* and *Home*); 8 individual documents contain the term (*Language*), 10 individual documents contain the term (*Arab*) and 7 individual documents contain the term (*Home*). The cosine similarity measures are calculated between query$_3$ terms and document terms according to equation (5) in section 4.3. From appendix table 6, we conclude that 79% of the collection matches query3. The total of terms' frequencies and the weights are equal to the total in appendix table 9. The top ranked documents which contain three terms are D$_{20}$, D$_{51}$, D$_{56}$, D$_{59}$, D$_{87}$, D$_{23}$, D$_{31}$, D$_{38}$, D$_{12}$, D$_{48}$, D$_{66}$, D$_{24}$, D$_{55}$, D$_{63}$, D$_{10}$ and D$_{62}$ (see figure 7.6).

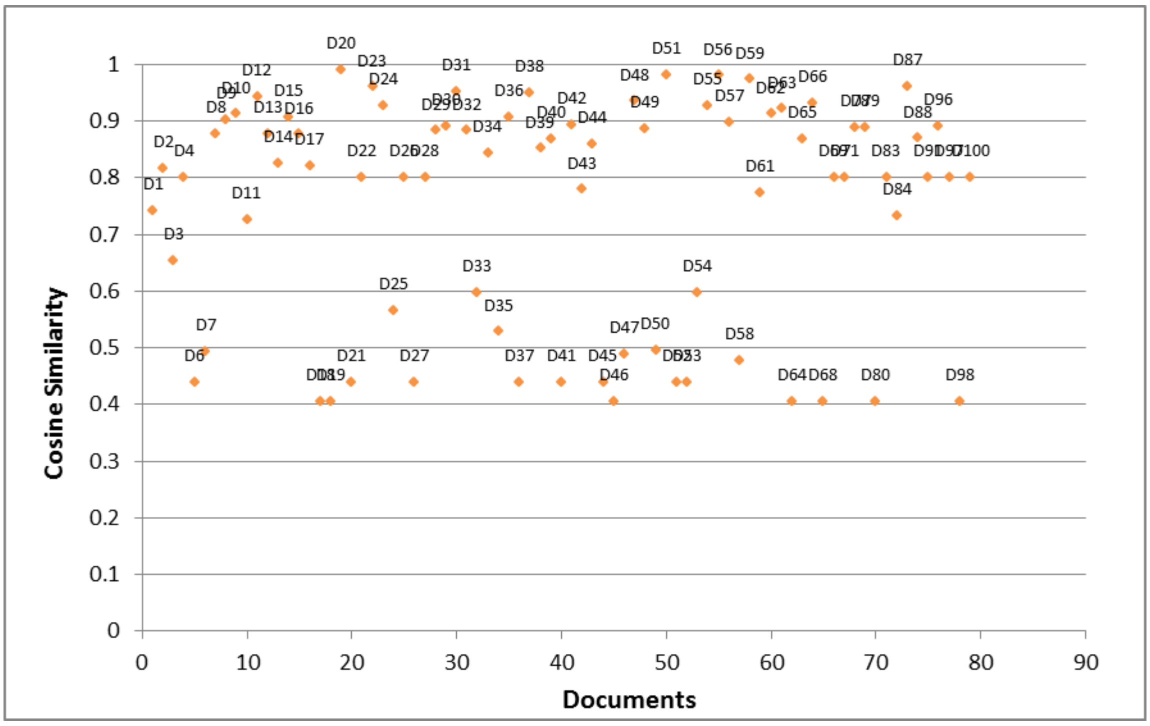


Figure 7.6. Similarity measures of query$_3$.

4. Query$_4$={استخدام الرياح في التنمية المستدامة - *Using wind in sustainable development*}, after stemming process has taken place, Query$_4$={خدم روح نمي دوم - *Use Develop Wind Sustain*}. In appendix table 7, query$_4$ has 4 terms. The (*Use*, *Develop*, *Wind* and *Sustain*) terms occurred once (TF=1). So, the inverse document frequency and the weight of query$_4$ were determined according to equations (3) and (4) in section 4.3. As a result of query$_4$ in appendix table 8, we have found 93 documents in total from the collection match query$_4$ terms. Twenty-six documents contain terms (*Use*, *Develop*, *Wind* and *Sustain*); 21 documents contain the terms (*Use*, *Develop* and *Wind*); 17 documents contain the terms (*Use*, *Develop* and *Sustain*); 3 documents contain the terms (*Use*, *Wind* and *Sustain*); 16 documents contain the terms (*Use* and *Develop*); 2 documents contain the terms (*Use* and *Wind*); 1 document contains the terms (*Develop* and *Wind*); 6 individual documents contain the term (*Use*); 1 individual document contains the term (*Develop*). The cosine similarity measures are calculated between query$_4$ terms and document terms according to equation (5) in section 4.3. From appendix table 8, we conclude that 93% of the collection matches query$_4$. The total of terms' frequencies and the weights are equal to total in appendix table 9. The top ranked documents which contain four terms are D$_{55}$, D$_{14}$, D$_{43}$, D$_{96}$, D$_{71}$ and D$_{18}$ (see figure 7.7).

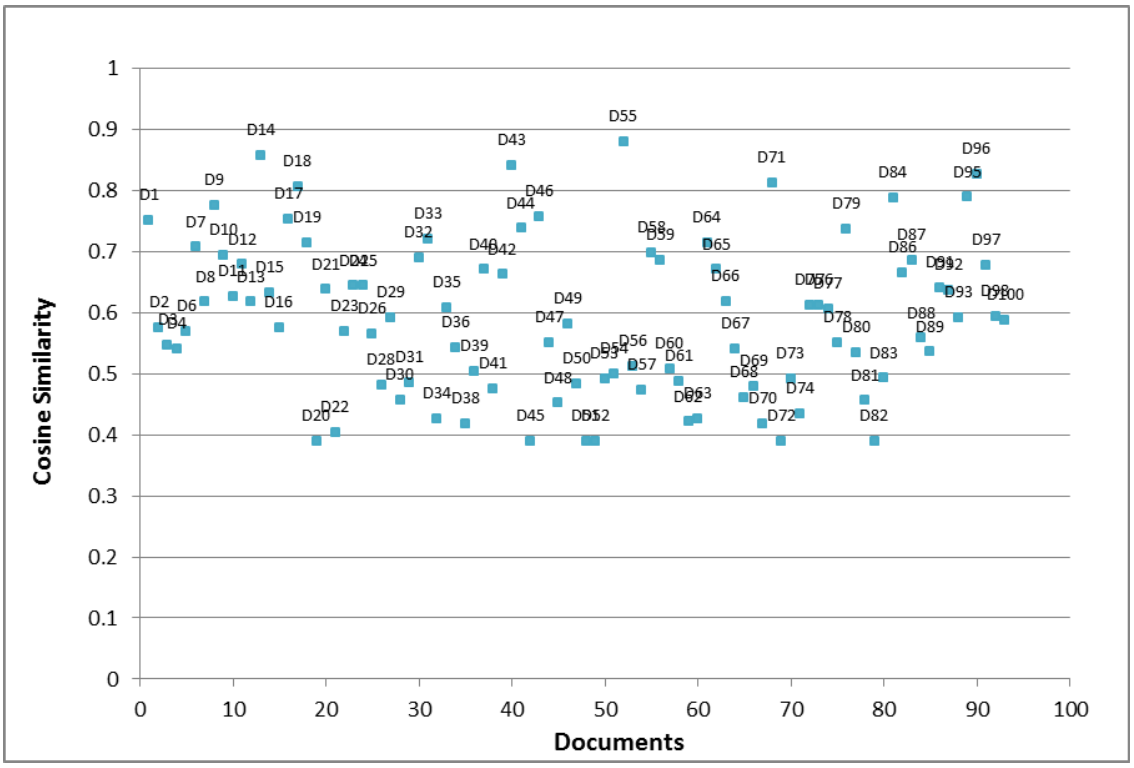


Figure 7.7. Similarity measures of query$_4$.

8. SECURING XML

XML data is written in plain text based on XML syntax. However, data processing requires a secure environment to achieve integrity, authentication, and non-repudiation. So, XML digital signature, encryption, and XKMS have been used to secure XML data. Following is a brief introduction to the techniques to create and verify the digital signature and the encryption.

8.1. Public Key Cryptography

Public key cryptography has been used to exchange data securely between users of an insecure network such as the Internet.

Each user has a pair of keys (public and private). The public key is available to anyone wants to communicate and the private key is protected. The keys in the pair is mathematically correlated to each other, as well as it's impossible to derive the private key from knowledge of the public key.

Based on public key cryptography, a document encrypted with the public key of a specific end user could only be decrypted by the correlated private key. Furthermore, a third party would unable to decrypt the document without the matching private key.

A sender's private key is used to create a digital signature for a document or data that to be signed and a recipient can verify the signature using sender's public key.

A certificate (X.509) has been used to ensure that the public key is belong to the sender (the third party couldn't claim to be the sender). A certificate authority has issued the certificate, which's a confirmation of the validity of the binding between the subject of the certificate and the corresponding public key. So, the other users are confident that the public key is related to the sender [79][80].

8.2. Digital Singnature

As number of documents, data and electronic transactions are increased. The digital signature has been used to make it possible to trust upon these transactions, documents, and data, because they signed by trusted source. The digital signature has been used as evidence of authenticity, data integrity and non-repudiation. Digital signing must provide a secure process and end-to-end privacy in data transmission.

Digital signature is created and verified using some algorithms, such as a Digital Signature Algorithm. The DSA allows an entity to authenticate the integrity of signed data and the identification of the signatory, because a digital signature must be created by an authorized person using his private key, as well as the corresponding public key used to verify the signature. Each signatory has paired public and private keys.

The digital signature schem consists of three algorithms, which are generating, signing and verfying according to public key cryptography. Different PKC schemes have been used to generate digital signature and implement data encryption, such as [77][78][79][81]:

1. RSA scheme.
2. DSA scheme.
3. ElGamal scheme.
4. ECDSA scheme.

However, a pair of a private key and a corresponding public key are mathematically generated and involved by these asymmetric schemes to generate a digital signature. The public key is published and can be shared with others. Private key only known by the user and must be kept safe and secret. The private key usually stored on encrypted hard drive, on smart cards, on a safe server or in a cloud. This key cryptography system shall comply with following mathematical properties [77][79]:

1. Encrypting a message with a private key, and decrypting the message within corresponding public key, will retrieve the original message.
2. Impossible to find out a private key using a corresponding public key.

Figure 8.1 illustrates the steps of signing a single document by a single user. The hash function is a one-way algorithm used toconvert a sequence of characters into a fixed length of values. Figure 8.2 describes the steps established by the receiver after received the signed document. Firstly, the encrypted hash is decrypted using the sender’s public key to get the original hash value. Secondly, the receiver generates a hash value from the received document. Finally, the hash values are compared. If they are equal, then the document is considered as authentic. If they don't match, it means the signature is not valid [79][81].

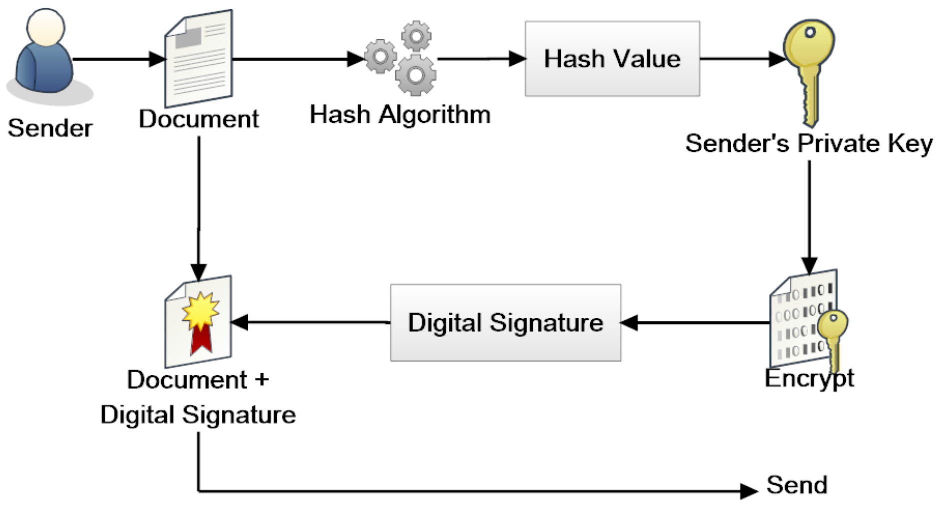


Figure 8.1. A single user signing a document.

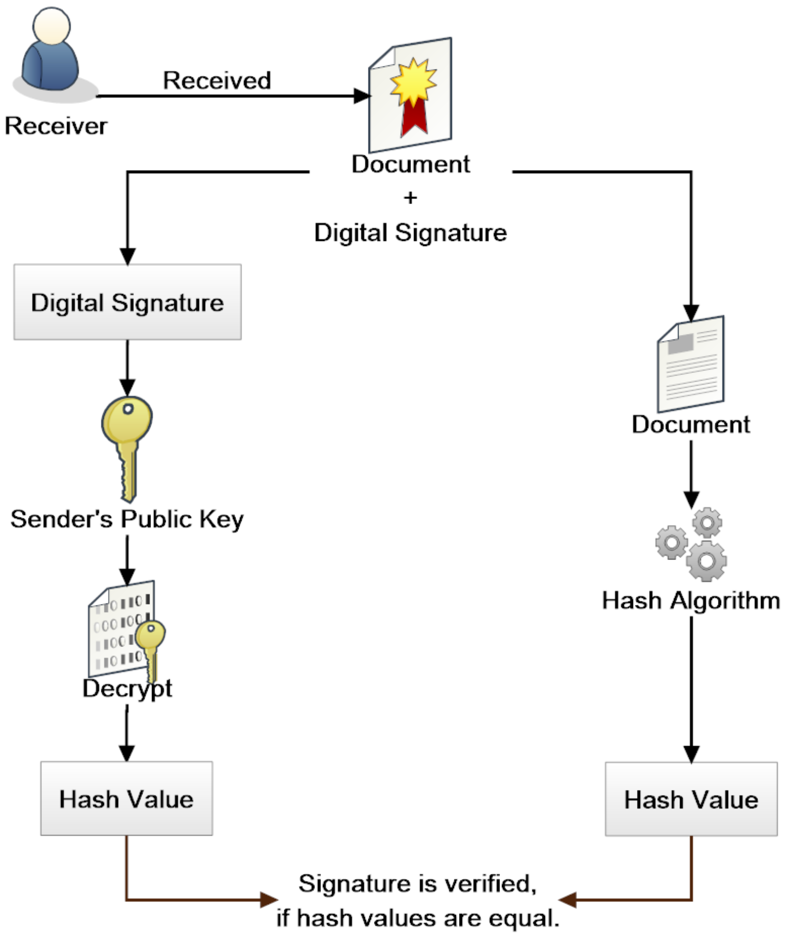


Figure 8.2. Verifying the signed document.

The digital signature guarantees authentication, non-repudiation, and integrity. These three information security properties are defined below [79][81][82]:

1. *Authentication*. Every signer is identified by PKC (private/public keys), there is no conflict within the other signers.
2. *Non-repudiation*. The signer couldn't deny performing the signing process because the signer private key was used for the encryption process.
3. *Integrity*. The signed document is associated with its corresponding digital signature. so, any document alteration will make the signature invalid as well as the signature cannot be copied from one document to another.

Some technologies such as certificates and block cipher are used by digital signature schemes to ensure the authentication. Furthermore, to ensure the confidentiality and privacy, the entire document could be encrypted using the receiver’s public key. on the receiver's side, the document could be decrypted using the receiver's private key, as well as time stamp could be added to the signature to trace the document during an active connection [79][82].

In a lot of cases, more than one signer is required to sign a document or even some parts of the document. Therefore, the Digital Multiple Signatures are invoked.

8.3. XML Digital Signature

An XML digital signature is a complex cryptographic object and it is already a digital signature including processing rules and syntax (defined by W3C recommendation) designed for use within XML documents and XML transactions. XML signature used to sign any data of any type, no matter the data is included in the XML document or not. XML signature guarantees the integrity, the document authentication, and the signer authentication service. The XML digital signature is created from a hash, such as SHA-1, over the canonical form of the digital signature manifest.

The XML signature has the ability to sign just a specific portion of the XML document instead of the whole document. This feature will make a single XML document has multiple signatures, such as each portion of XML document will be signed by different parties (every one will sign only the portions that belong to himself), as well as its important to guarantee the integrity of these portions and allow the others to change and update theirs, for example, a signed XML document sent to an employee for completion. If the signature was over the whole XML document, any change by the employee will invalidate the original signature. The XML signature can sign any type of resources, such as HTML data, JPEG binary data, XML-encoded data, a specific portion of an XML file, ... etc. [80].

Signature verification needs that the data object that was signed be available and accessible. The XML digital signature itself will clearly indicate the references, which include the location of the original signed object. Theses references are [80]:

1. URI included within the XML digital signature.
2. XML digital signature resides within the object resource itself (sibling signature).
3. The object source embedded within the XML digital signature (parent signature).
4. XML digital signature embedded within the object source (child signature).

8.3.1. The Structure of an XML Signature

Figure 8.3 shows the XSD of the XML digital signature and listing 8.1 describes the structure of an XML digital signature, where "*" means zero or more, "+" means one or more and "?" means zero or one occurrence. The main concepts of XML digital signature structure are [79][80]:

1. Each signed resource has a `<Reference>` element that identified by URI attribute.
2. A `<Transform>` element specifies a processing steps based on transform algorithm that applied to the resource's content which referenced by the URI attribute.
3. A `<DigestValue>` element contains the value of the digest (hash) of the resource which referenced by the URI attribute.
4. A `<SignatureValue>` element contains the value of the encrypted digest of a `<SignedInfo>` element.
5. A `<KeyInfo>` element has the key which used to validate the XML digital signature, such certificates, and algorithms.

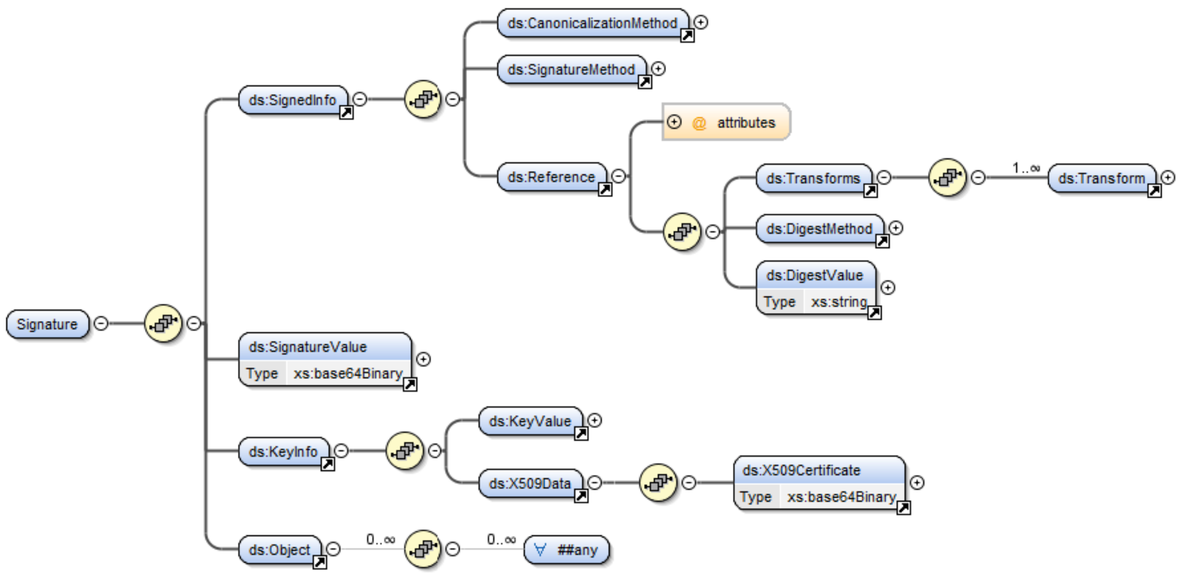


Figure 8.3. XSD of an XML Digital Signature.

```
<Signature>
   <SignedInfo>
     <CanonicalizationMethod>
     <SignatureMethod>
     (<Reference (URI=)?>
        (<Transforms>)?
        <DigestMethod>
        <DigestValue>
     </Reference>)+
   </SignedInfo>
   <SignatureValue>
   (<KeyInfo>)?
   (<Object>)*
</Signature>
```

Listing 8.1. XML Digital Signature structure [79].

There are three types of the XML digital signature (see figure 8.4) [83]:

1. *Enveloped XML Signature*. In enveloped XML digital signature, the XML signature is embedded within the signed document.
2. *Enveloping XML Signature*. In enveloping XML digital signature, the signed document is embedded within the XML digital signature structure.
3. *Detached XML Signature*. In detached XML digital signature, the signed entities are separated from the signature structure. The signed entities can be remote XML

documents, remote data or located elsewhere in the same document as the XML digital signature.

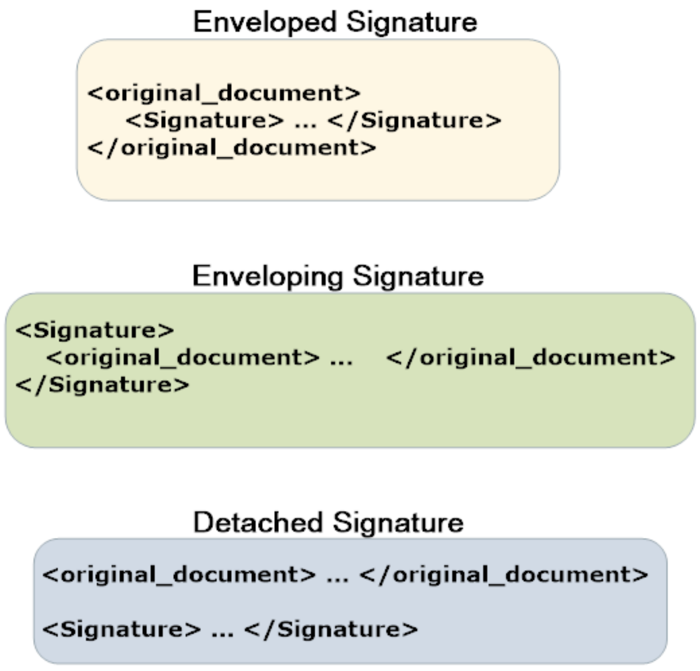

Enveloped Signature
<original_document>
<Signature> ... </Signature>
</original_document>
Enveloping Signature
<Signature>
<original_document> ... </original_document>
</Signature>
Detached Signature
<original_document> ... </original_document>
<Signature> ... </Signature>

Figure 8.4. Types of XML digital signature.

8.3.2. Creation of an XML Digital Signature

The specification of an XML digital signature is divided into syntax and processing rules. Moreover, the following steps must establish to successfully create the XML digital signature [79][80]:

1. Select the resources that need to be signed. An URI is used to identify different resources, for example:

   - The URI “http://www.raiffeisenbank.rs/retail-services.795.html” is the reference of an HTML page on the Internet.
   - The URI “https://upload.wikimedia.org/wikipedia/en/1/19/Statcounter_logo.gif” is the reference of a GIF image on the Internet.
   - The URI “http://www.w3schools.com/xsl/books.xml” is the reference of an XML file on the Internet.

- The URI "http://www.w3schools.com/xsl/books.xml#author" is the reference of a specific element in an XML file on the Internet.

2. Calculate the digest value for each resource. In XML digital signatures, each referenced resource is determined by a `<Reference>` element and its digest value is stored in a `<DigestValue>` child element. The `<DigestMethod>` element sets the algorithm which used to calculate the digest value (see listing 8.2).

```
<Reference URI=" http://www.raiffeisenbank.rs/retail-services.795.html">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
</Reference>

<Reference URI=" http://www.w3schools.com/xsl/books.xml">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
</Reference>
```

Listing 8.2. The degist calculations.

3. Collect the reference elements. Collect the `<Reference>` elements including corresponding digest values within a `<SignedInfo>` element. The `<CanonicalizationMethod>` element sets the algorithm which used to canonize the `<SignedInfo>` element. The `<SignatureMethod>` element sets the algorithm which used to generate the signature value (see listing 8.3).

```
<SignedInfo Id="si">
<CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
<Reference URI=" http://www.raiffeisenbank.rs/retail-services.795.html">
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
</Reference>

<Reference URI=" http://www.w3schools.com/xsl/books.xml">
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
</Reference>
</SignedInfo>
```

Listing 8.3. Collect the reference and digest elements.

4. Signing. Calculate the digest of the `<SignedInfo>` element, then sign this digest and place the signature value in a `<SignatureValue>` element (see listing 8.4).

```
<SignatureValue>MC0E~LE=</SignatureValue
```

Listing 8.4. Signature value.

5. Add key information. The key information is placed in a `<KeyInfo>` element. The keying information has the X.509 certificate for the sender, which already includes the public key which used for signature verification (see listing 8.5).

```
<KeyInfo>
<X509Data>
<X509SubjectName>CN=Hesham,O=SoftWareInc.,ST=TRIPOLI,C=LY
</X509SubjectName>
<X509Certificate>MIID5jCCA0+gA...lVN</X509Certificate>
</X509Data>
</KeyInfo>
```

Listing 8.5. Key information.

6. Finally, create a signature element. To create an XML digital signature element, the `<SignedInfo>`, `<SignatureValue>`, and `<KeyInfo>` elements must be placed in an element named `<Signature>` (see listing 8.6).

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo Id="si">
<CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>

<Reference URI=" http://www.raiffeisenbank.rs/retail-services.795.html">
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
</Reference>

<Reference URI=" http://www.w3schools.com/xsl/books.xml">
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
</Reference>

</SignedInfo>

<SignatureValue>MC0E~LE=</SignatureValue>

<KeyInfo>
<X509Data>
<X509SubjectName> CN=Hesham,O=SoftWareInc.,ST=TRIPOLI,C=LY
</X509SubjectName>
<X509Certificate>MIID5jCCA0+gA...lVN</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
```

Listing 8.6. Enclose in a Signature element.

8.3.3. Validation of the XML Digital Signature

In digital signature validation, there is no need to generate a key pair, because it already attached in the key information element. Moreover, following steps describes briefly how to verify the XML digital signature [79][80][83][84]:

1. First of all, the digest algorithm in the `<SignatureMethod>` element is used to recalculate the digest value of the `<SignedInfo>` element. Furthermore, the public key is used to verify that the value of the `<SignatureValue>` element is true for the digest value of the `<SignedInfo>` element.
2. Finally, by recalculating the digest values of the references which attached to the `<SignedInfo>` element. After that compare these digest values to the

corresponding `<DigestValue>` element within each `<Reference>` element. If this comparison passes, then the XML digital signature is valid.

XML digital signature standard defines many transformations that required for representation of signing and validation of XML digital signature, such as [79][83]:

- Canonicalization
- Base64 decoding transformer
- XPath filtering
- Enveloped transformer
- C14N transformer

8.4. XML Document Encryption

Encryption is used to implement privacy and confidentiality. XML encryption is W3C recommendation and specifies how to encrypt and decrypt data. XML Encryption provides end-to-end security and privacy for any participants or applications that require secure environment to exchange data. XML encryption illustrates a process for encrypting data and represents the result in XML syntax. The data may be an arbitrary data (an XML document), an XML element, or an XML element content. The encrypting data is an XML Encryption element which references the cipher data. Based on standard of XML Encryption, there are two ways to encrypt XML documents as following [85][86]:

1. Symmetric encryption. In this case symmetric algorithm such as AES has been used. The sender and the receiver share the same key that used in encryption and decryption processes of the XML document.
2. Asymmetric encryption. In this case, the sender encrypts the XML document using the receiver's public key such as RSA public/private key pair. Furthermore, the receiver decrypts the encrypted XML document using his private key.

8.4.1. Structure of XML Encryption

Figure 8.5 shows the XSD structure of XML encryption data and listing 8.7 describes the structure of XML Encryption data which's represented in `<EncryptedData>` element, where "?" denotes zero or one occurrence; "+" denotes one or more occurrences; "*" denotes zero or more occurrences; "|" denotes a selection [85].

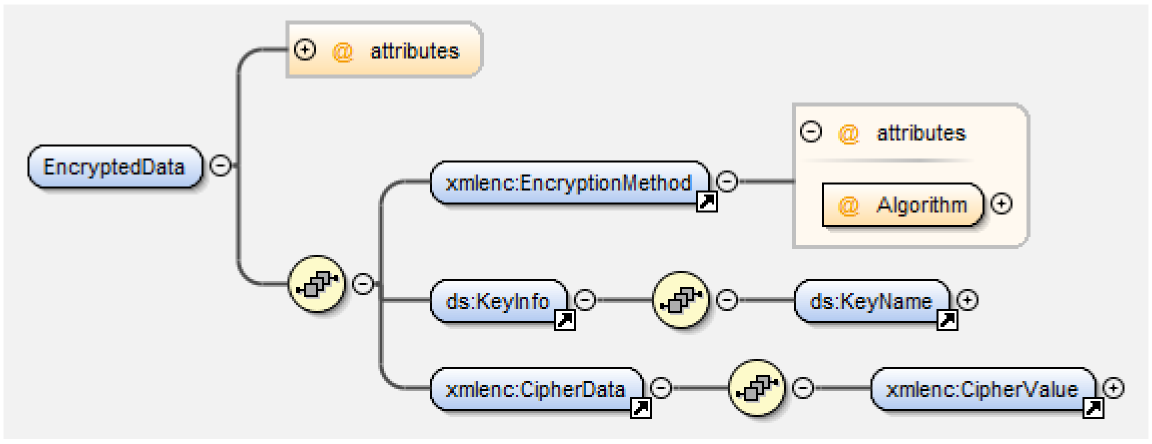


Figure 8.5. XSD of structure of an XML encrypted data.

```
<EncryptedData Id? Type? MimeType? Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?
    <AgreementMethod>?
    <ds:KeyName>?
    <ds:RetrievalMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue> | <CipherReference URI?>
  </CipherData>
  <EncryptionProperties>?
</EncryptedData>
```

Listing 8.7. Structure of XML Encryption data [85].

8.4.2. The XML Encryption Process

To encrypt any data within any XML document, the following steps must take into account [85][86][87]:

1. First of all, select an appropriate algorithm for the encryption process.
2. Obtain the encryption key (symmetric or asymmetric).
3. Encrypt the data (element or element's content), and place the result into `<CipherData>` element.
4. Place the `<CipherData>` element into the `<EncryptedData>` element.

The encryption has been used to encrypt a specific XML element, a specific XML element’s content, XML element’s data, arbitrary data and XML documents, as well as super encryption (encrypting encrypted data).

Different types of XML encryption and processes have been applied to a sample of an XML document. This sample is listed in listing 8.8. This will be discussed in more details below [85][86][87]:

```
<?xml version="1.0"?>
<Info xmlns="http://demo.org/info">
  <Name>Hesham Elzentani</Name>
  <IdCard>
    <CardNumber>123456789</CardNumber>
    <Nationality>Libya</Nationality>
    <Validation>04/02</Validation>
  </IdCard>
</Info>
```

Listing 8.8. Sample of an XML document that used in encryption.

1. *Encryption of an XML element*. XML Encryption is used to replace the target XML element with an `<EncryptedData>` element. The `<EncryptedData>` element may contain sub-elements that have information about the keys and processes used in encryption. XML document may contain multiple encrypted elements and any element could be encrypted multiple times. Listing 8.9 shows a single encrypted element (`<IdCard>`) of the XML document which listed in listing 8.8.

```
<?xml version="1.0"?>
<Info xmlns="http://demo.org/info">
  <Name>Hesham Elzentani</Name>
  <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
                 xmlns="http://www.w3.org/2001/04/xmlenc#">
    <CipherData>
      <CipherValue>AA35C15B6</CipherValue>
    </CipherData>
  </EncryptedData>
</Info>
```

Listing 8.9. Encryption of an XML element

2. *Encryption of an XML element’s content (elements of an element)*. Here, XML encryption is used to replace the target XML elements of a specific element with the `<EncryptedData>` element. Listing 8.10 shows the encryption of the XML

elements (`<CardNumber>`, `<Nationality>` and `<Validation>`) of the specific XML element (`<IdCard>`) in the XML document sample, which listed in listing 8.8.

```
<?xml version="1.0"?>
<Info xmlns="http://demo.org/info">
  <Name>Hesham Elzentani</Name>
  <IdCard>
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
                 Type="http://www.w3.org/2001/04/xmlenc#Content">
      <CipherData>
        <CipherValue>AA35C15B6</CipherValue>
      </CipherData>
    </EncryptedData>
  </IdCard>
</Info>
```

Listing 8.10. Encryption of XML elements of an element.

3. *Encryption of an XML element's content (data)*. In this case, XML encryption is used to encrypt the character data of a specific XML element. Furthermore, in listing 8.11, consider that all the information except the actual card number can be in the clear, including the fact that the card number element exists. Both `<IdCard>` and `<CardNumber>` elements are in the clear, but the character data content of `<CardNumber>` element is encrypted.

```
<?xml version="1.0"?>
<Info xmlns="http://demo.org/info">
  <Name>Hesham Elzentani</Name>
  <IdCard>
    <CardNumber>
      <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
                     Type="http://www.w3.org/2001/04/xmlenc#Content">
        <CipherData>
          <CipherValue>AA35C15B6</CipherValue>
        </CipherData>
      </EncryptedData>
    </CardNumber>
    <Nationality>Libya</Nationality>
    <Validation>04/02</Validation>
  </IdCard>
```

Listing 8.11. Encryption of an XML element's data character.

4. *Encryption of arbitrary data and XML documents*. In this case, all information is encrypted, the whole XML document or arbitrary data is encrypted as an octet sequence (see listing 8.12).

```
<?xml version="1.0"?>
<EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
               MimeType="text/xml">
  <CipherData>
    <CipherValue>AA35C15B6</CipherValue>
  </CipherData>
</EncryptedData>
```

Listing 8.12. Encryption of arbitrary data.

5. *Super-encryption (encrypting encrypted data).* The XML document may include zero or more `<EncryptedData>` elements. In general, the `<EncryptedData>` element cannot be a parent or a child of another `<EncryptedData>` element. Moreover, the actual data encrypted can be of any type, as well as the `<EncryptedData>` element and the `<EncryptedKey>` element. In super-encryption of the `<EncryptedData>` element or the `<EncryptedKey>` element, one must encrypt the entire element. Listing 8.14 represents encryption of original data with `Id='END1'`, where in listing 8.15, the content of `<CipherValue>` element has "New encrypted data" of base64 encoding of the encrypted octet sequence resulting from encrypting the `<EncryptedData>` element with `Id='END1'`.

```
<?xml version="1.0"?>
<in:Info xmlns:in="http://demo.org/info">
  <EncryptedData Id="END1"
                 xmlns="http://www.w3.org/2001/04/xmlenc#"
                 Type="http://www.w3.org/2001/04/xmlenc#Element">
    <CipherData>
      <CipherValue>Original encrypted data</CipherValue>
    </CipherData>
  </EncryptedData>
```

Listing 8.13. Original encrypted data.

```
<?xml version="1.0"?>
<in:Info xmlns:in="http://demo.org/info">
  <EncryptedData Id="END2"
                 xmlns="http://www.w3.org/2001/04/xmlenc#"
                 Type="http://www.w3.org/2001/04/xmlenc#Element">
    <CipherData>
      <CipherValue>New encrypted data</CipherValue>
    </CipherData>
  </EncryptedData>
</in:Info>
```

Listing 8.14. Super-encryption of original encrypted data in listing 8.13.

8.4.3. The XML Decryption Process

To decrypt any data within `<EncryptedData>` element in any XML document, the following steps must take into account [85][86][87]:

1. First of all, determine the algorithm and the parameters used in the encryption process, as well as the `<KeyInfo>` element.
2. Find the key which placed in the `<KeyInfo>` element. If the key is encrypted, decrypt it with the corresponding key.
3. Decrypt the data which placed in the `<CipherData>` element and decode base64 to retrieve the text value. If `<CipherReference>` element is found, retrieve the encrypted octet sequence using transforms according to the URI.
4. The encrypted data is decrypted using the algorithm, parameters and key information which are obtained from first step. No matter if it's element, element's content or document. Finally, replace the `<EncryptedData>` element with the decrypted data (the decrypted data must be in XML format).

8.5. Simple Object Access Protocol

In the cyber world, SOAP has been used in transactions such as commerce transactions, as well as WSDL has been used in Web services. Both SOAP and WSDL are structured based on XML syntax. For security issues, XML digital signature and XML encryption can be used to ensure confidentiality.

SOAP is W3C recommendation protocol written in XML format that used to send and receive structured messages between participants, systems or applications in computer networks. SOAP is extensible, independent and has the ability to use it over any transport protocol such as HTTP and TCP [88][89].

Furthermore, the SOAP is used by web applications to communicate with each other over the Internet. A SOAP sender sends a SOAP message to a SOAP receiver (SOAP nodes), such as query request and its response [88].

A SOAP message is considered as an XML document. The framework of the SOAP messaging are [89]:

1. Processing Model.
2. Extensibility Model.
3. Protocol Binding Framework.
4. Message Construct.

8.5.1. SOAP structure

A SOAP message is structured from three main elements as shown in figure 8.6 and two optional elements as followed:

1. `<soap: Envelope>` element as a parent.
2. `<soap: Header>` element as a child that contains header information and includes an optional `<wsse: security>` element which contains security information.
3. `<soap: Body>` element as a child that contains request and response information and includes an optional `<soap: Fault>` element which contains error information.

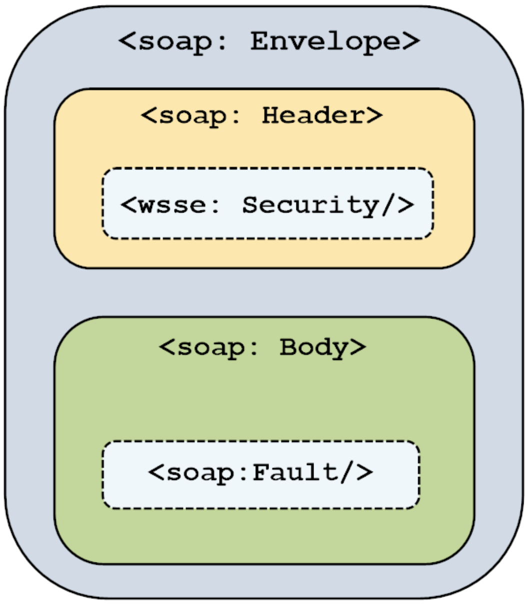


Figure 8.6. SOAP structure.

The elements of SOAP message are declared in the namespace of the `<soap: Envelope>` element as “http://www.w3.org/2003/05/soap-envelope/”, encoding and data types as “http://www.w3.org/2003/05/”, (see listing 8.15).

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
   soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
   <soap:Header/>
   <soap:Body>
     <soap:Fault/>
   </soap:Body>
</soap:Envelope>
```

Listing 8.15. XML structure of SOAP message.

8.5.2. Creation of SOAP message

There are some syntax rules must follow up to create a successful SOAP message and to exchange information between two terminals. These rules are below:

1. A message must encode correctly using XML syntax.
2. A message must have the envelope and encoding namespaces.
3. A DTD is excluded from a SOAP message.
4. XML processing instructions are excluded from a SOAP message.

For clearance, listing 8.16 and listing 8.17 showed an example of request and response SOAP messages between two terminals over the Internet.

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
   soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
   <soap:Body>
      <m:GetIdNo xmlns:m="http://www.example.com/IdNo">
         <m:Name>Hesham Elzentani</m:Name>
      </m:GetIdNo>
   </soap:Body>
</soap:Envelope>
```

Listing 8.16. Example of SOAP message in request form.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soap:Body>
  <m: GetIdNoResponse xmlns:m="http://www.example.com/IdNo">
    <m:IdNo>123456</m:IdNo>
  </m: GetIdNoResponse>
</soap:Body>
</soap:Envelope>
```

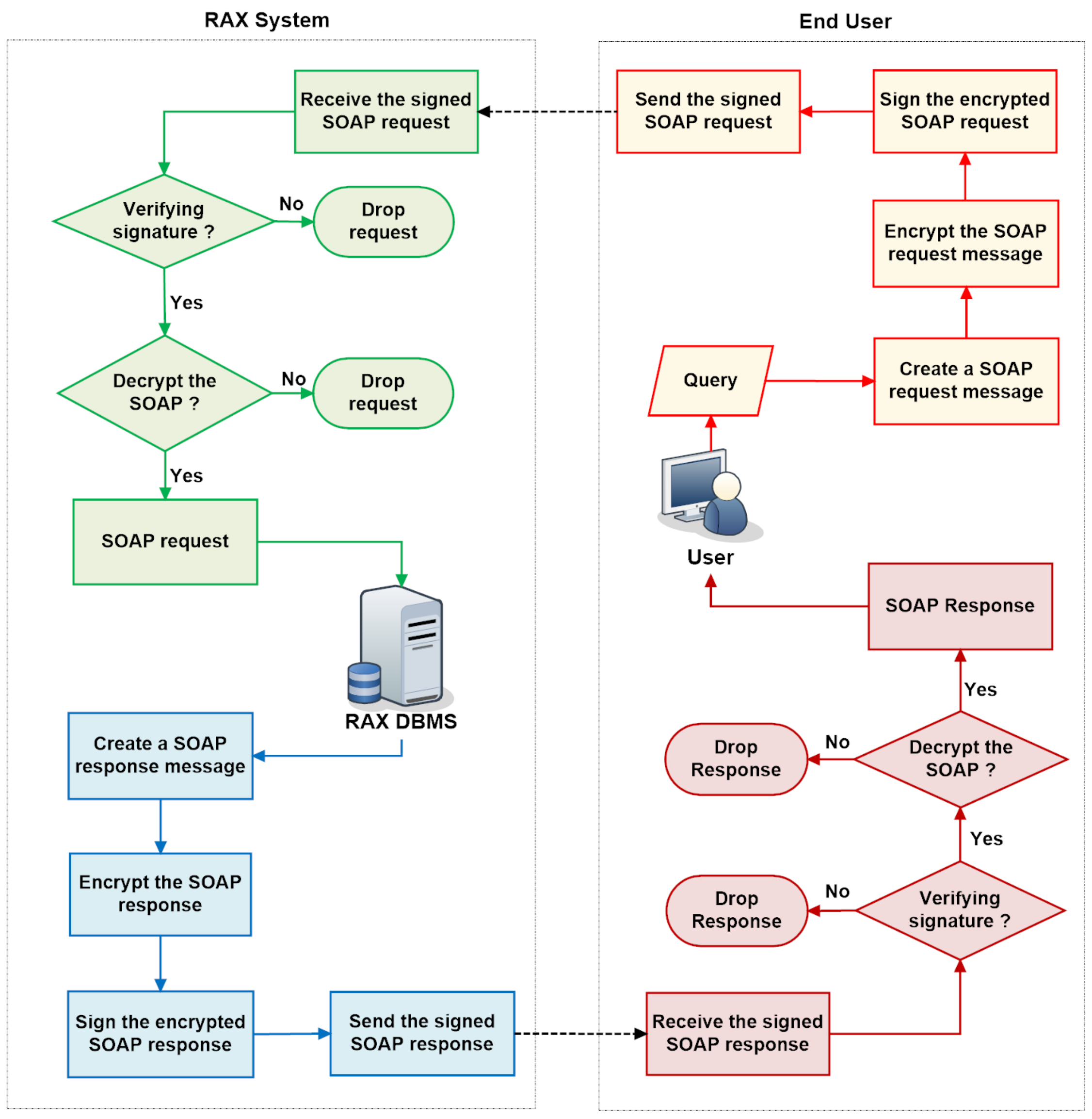
Listing 8.17. Example of SOAP massage in response form.

9. SECURING THE RAX SYSTEM

In this section, we will discuss a method of how making the RAX system (which mentioned in section 7) working in a secure, private and confidential environment.

9.1. The Securing Model of the RAX System

This thesis proposes a survey of how making a combination between XML security (XML digital signature and XML encryption) and the SOAP message to create a secret environment between an end user and the RAX system model. Figure 9.1 shows the securing model of the RAX system.



RAX System
End User
Receive the signed SOAP request
Verifying signature ?
No
Drop request
Yes
Decrypt the SOAP ?
No
Drop request
Yes
SOAP request
RAX DBMS
Create a SOAP response message
Encrypt the SOAP response
Sign the encrypted SOAP response
Send the signed SOAP response
Send the signed SOAP request
Sign the encrypted SOAP request
Encrypt the SOAP request message
Create a SOAP request message
Query
User
SOAP Response
Yes
Drop Response
No
Decrypt the SOAP ?
Yes
Drop Response
No
Verifying signature ?
Receive the signed SOAP response

Figure 9.1. Securing model the RAX system.

9.1.1. How the Securing model Works

The securing model of the RAX system (see figure 9.1) consists of four stages, which are SOAP request creation, SOAP request verification, SOAP response creation and SOAP response verification. These stages will be described in more details below:

1. *Creation of a SOAP request message*. The end user's query in our model is the basis of the SOAP request creation process. Following is the steps of how creation of a SOAP request message are (see figure 9.2):

   - Convert the user's query to a SOAP request message that contains header and body elements based on XML syntax.
   - Encrypt the SOAP request message according to the RAX's information key to ensure the integrality and the confidentiality (see figure 9.6).
   - Create XML digital signature according to the user's key information and sign the encrypted SOAP request message (see figure 9.6).
   - Finally, forward the signed SOAP request message.

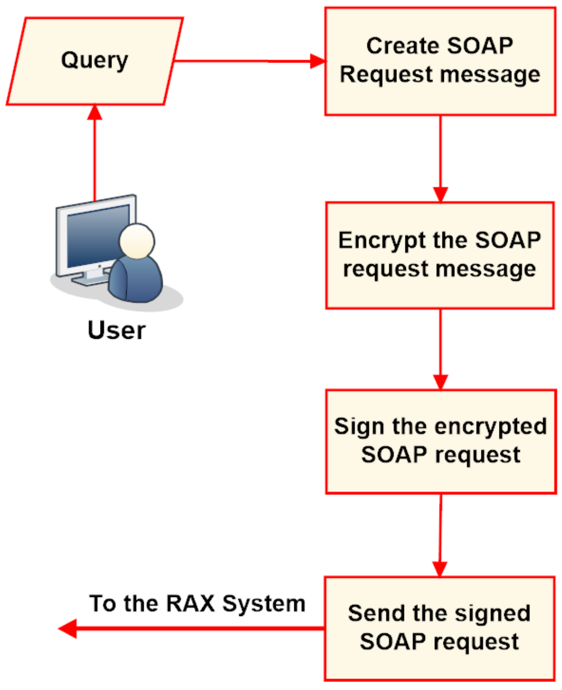

Query
Create SOAP Request message
User
Encrypt the SOAP request message
Sign the encrypted SOAP request
To the RAX System
Send the signed SOAP request

Figure 9.2. Creation of SOAP message based on user's query.

2. *Verification of the SOAP request message*. Following is the steps of how verification of the signed and encrypted SOAP request message are established (see figure 9.3):

   - Receive the signed SOAP request message.
   - Verify XML digital signature. Go to the next step if the signature is valid, or drop the SOAP request message if the signature is invalid.
   - Decrypt the encryption of the signed SOAP request message. If the decryption process succeeds, go further to the next step or drop the SOAP request message if the decryption process failed.
   - Finally, deliver the SOAP request message to the RAX system.

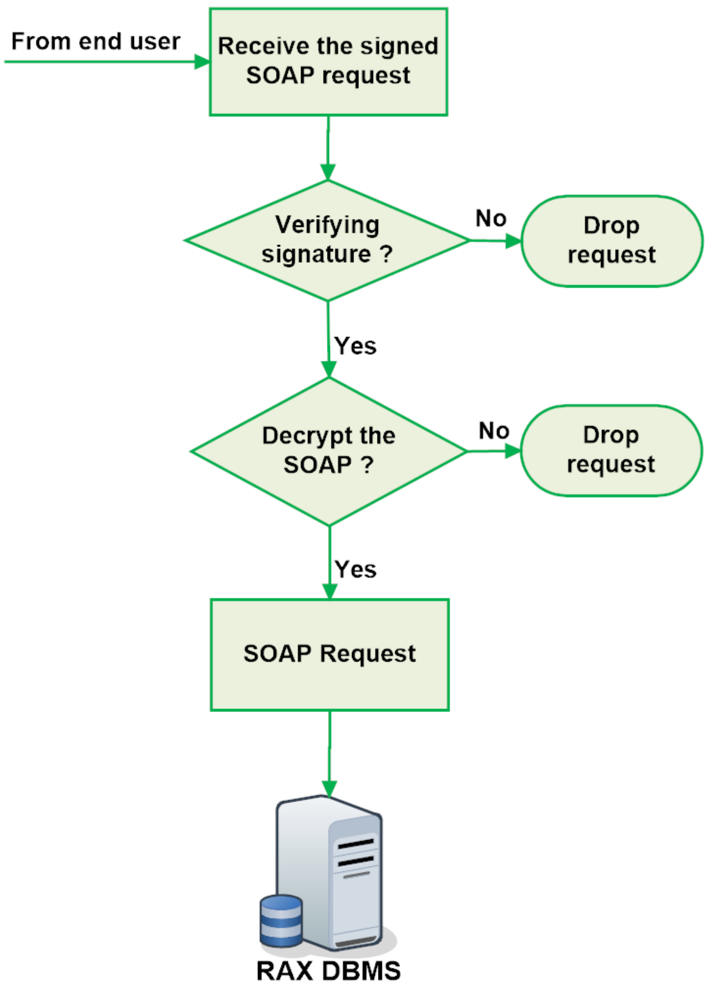


Figure 9.3. Verifying the encrypted SOAP request message.

3. *Creation of the SOAP response message*. Following is the steps of how the SOAP response message is created and authenticated (see figure 9.4):

   - The RAX system will answer the user's query. Based on this answer, the SOAP response message is created (in XML format).
   - In this step, the encryption process will take a place according to the user's information key to encrypt the SOAP response message (see figure 9.6).
   - Create XML digital signature according to the RAX's information key and sign the encrypted SOAP response message (see figure 9.6).
   - Finally, the signed SOAP message will be sent to the end user.

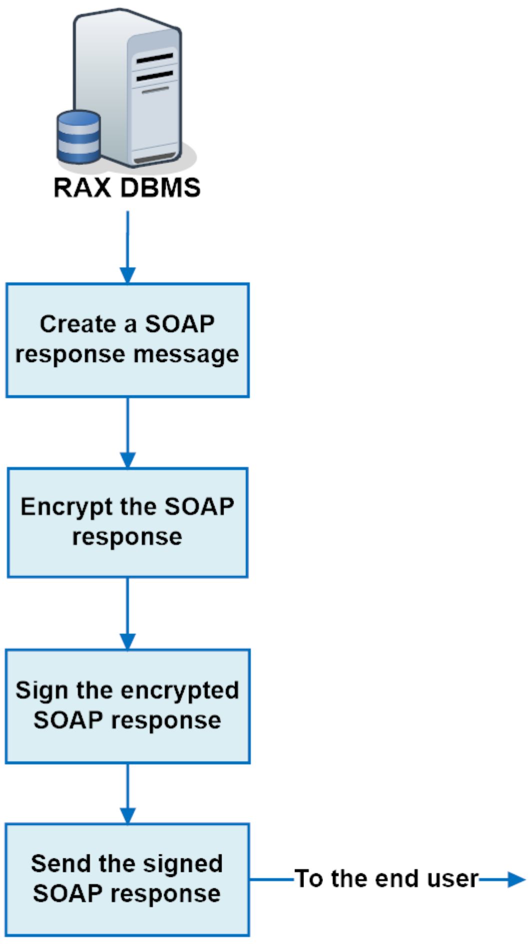

RAX DBMS
Create a SOAP response message
Encrypt the SOAP response
Sign the encrypted SOAP response
Send the signed SOAP response
To the end user

Figure 9.4. Creation and authentication of the SOAP response message.

4. *Verification of the SOAP response message*. Following is the steps of how verification of the encrypted SOAP response message are established (see figure 9.5):

- Receive the encrypted SOAP response message.
- Decrypt the encryption of the SOAP request message. If the decryption process succeeds, go further to the next step or drop the SOAP response message if the decryption process failed.
- Verify XML digital signature. Go to the next step if the signature is valid, or drop the SOAP response message if the signature is invalid.
- Finally, deliver the SOAP response message to the end user.

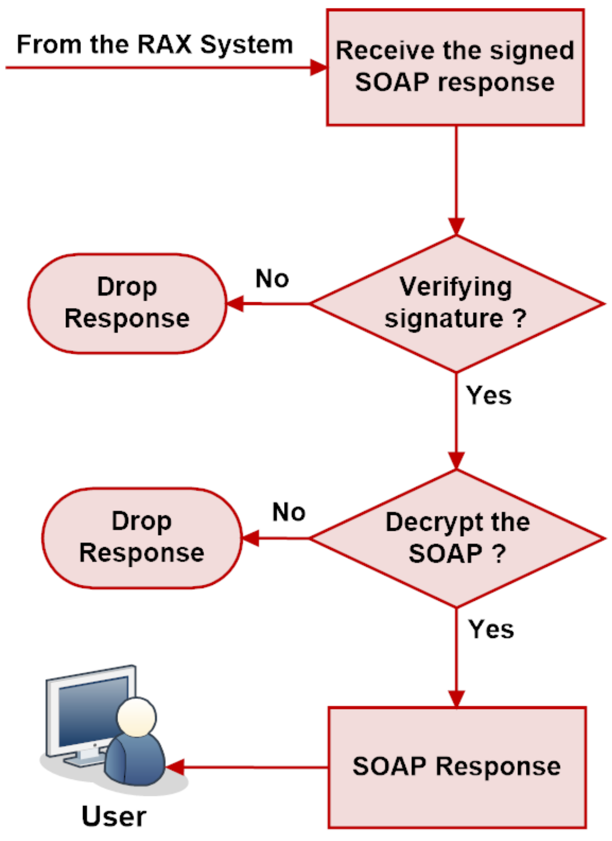


Figure 9.5. Verifying the encrypted SOAP response message.

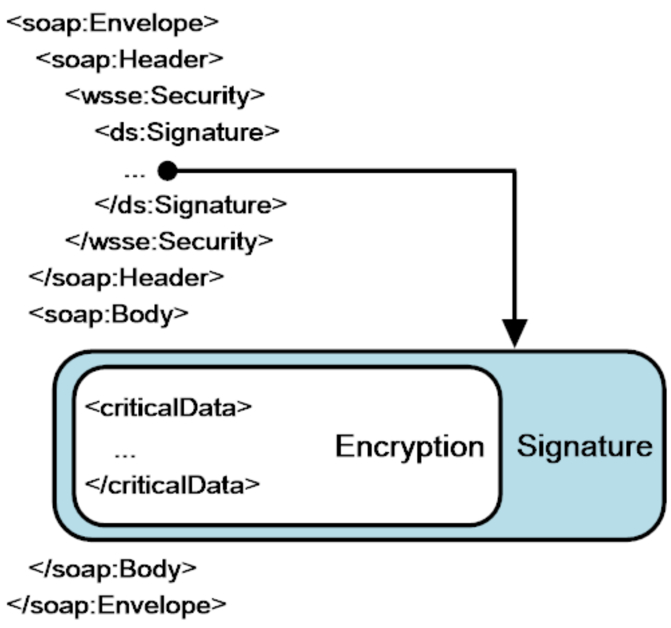


Figure 9.6. the structure of the signed encrypted SOAP message.

9.2. Attacks and Countermeasures

In this section, we are going to discuss the most important attacks that will affect the security issues of the SOAP message (XML digital signature and XML data encryption attacks) and present the countermeasures for each attack.

9.2.1. XML Signature Attacks and countermeasures

Following are the most important XML digital signature attacks including their countermeasures:

1. *Attack and countermeasure of an XML signature wrapping*. In the XML signature wrapping attack, the attacker will inject a faked element within a structure of a SOAP message. However, the fake element does not affect the XML digital signature element, because the web service which used to process the SOAP message will verify the signature in a separated process from the invoked application. So the signature is valid and covers the corresponding element which used in signature verification. In contrast, the faked element is processed by an application logic. Figure 9.7 describes an example of an XML signature wrapping attack of the SOAP message [90][91][92].

The countermeasure of this attack can take a place according to the following techniques:

- The sender and the receiver must improve their security policies. This technique was presented in [90].
- Validating the XML schema has been used as the countermeasure to this attack. This technique was presented in [93].
- The inline approach is used to fix the position of the signed element in the structure of the SOAP message, so any change or movement to the signed element will invalidate the XML signature. This approach was described in [94].
- XML watermarking technique. It uses the constraints of the XML schema to fix its structure using the watermark technique [95].
- Time stamp has been used to specify the expiry time of the SOAP message. So, the expired message is already rejected.

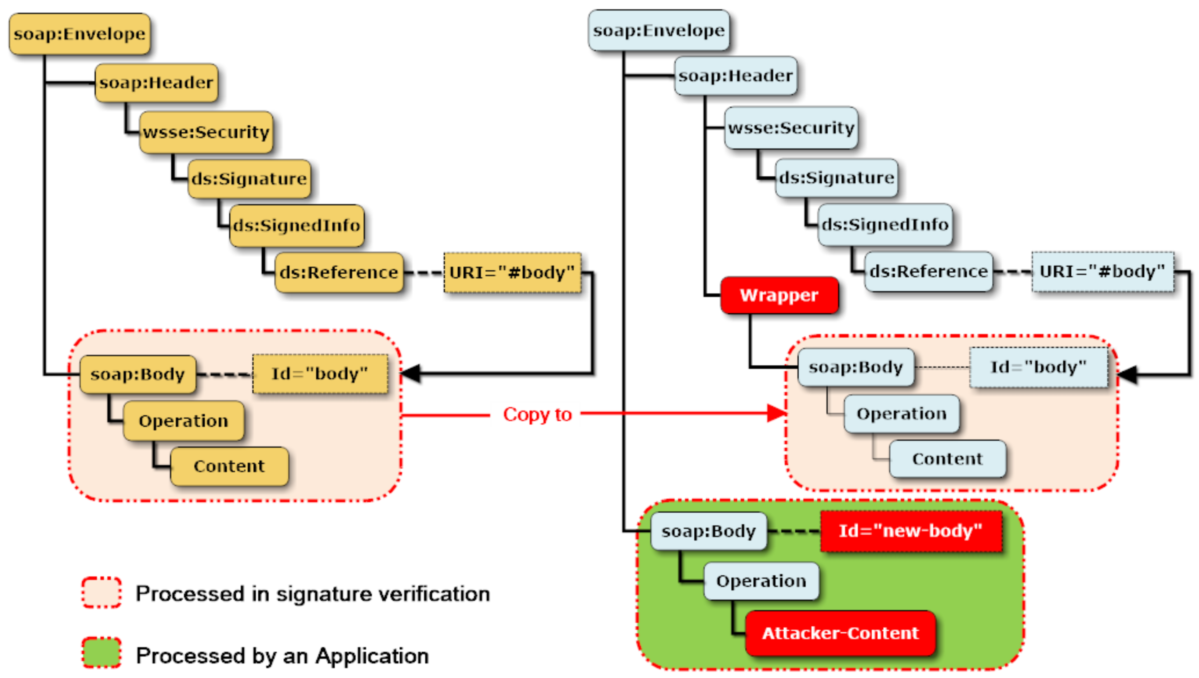


Figure 9.7. An XML signature wrapping attack of a SOAP message.

2. *Attack and countermeasure of XPath Based XML Signature*. XML digital signature sometimes uses XPath expressions to refer to the position of the signed element in the SOAP message [96]. The XPath expression is related to the reference element, which

protected by the XML signature. Furthermore, the signature element must be kept protected when an attacker tried to attack the SOAP message.

In this attack, the attacker used descendant axis specifiers (Depth-First and Breadth-First) to attack the XPath of the SOAP message (see figure 9.8). Furthermore, the XPath of the SOAP message which haven't a descendant-* axis specifier can be attacked using an XPath expression depending on the attribute of the signed element (see figure 9.9) [92]. So, to have a succeed attack, the attacker must implement the both attacks (figure 9.8 and figure 9.9) together as following:

- The attacker must locate a prober location to the body of the SOAP message based on descendant axis specifier.
- The attacker must modify the attributes in both, signed and attacked elements.

The countermeasure of this attack can take a place according to the FastXPath, because the FastXPath syntax denies the usage of descendant axis. The FastXPath uses SAX/StAX parser to parse the SOAP message and supports just forward axis selection.

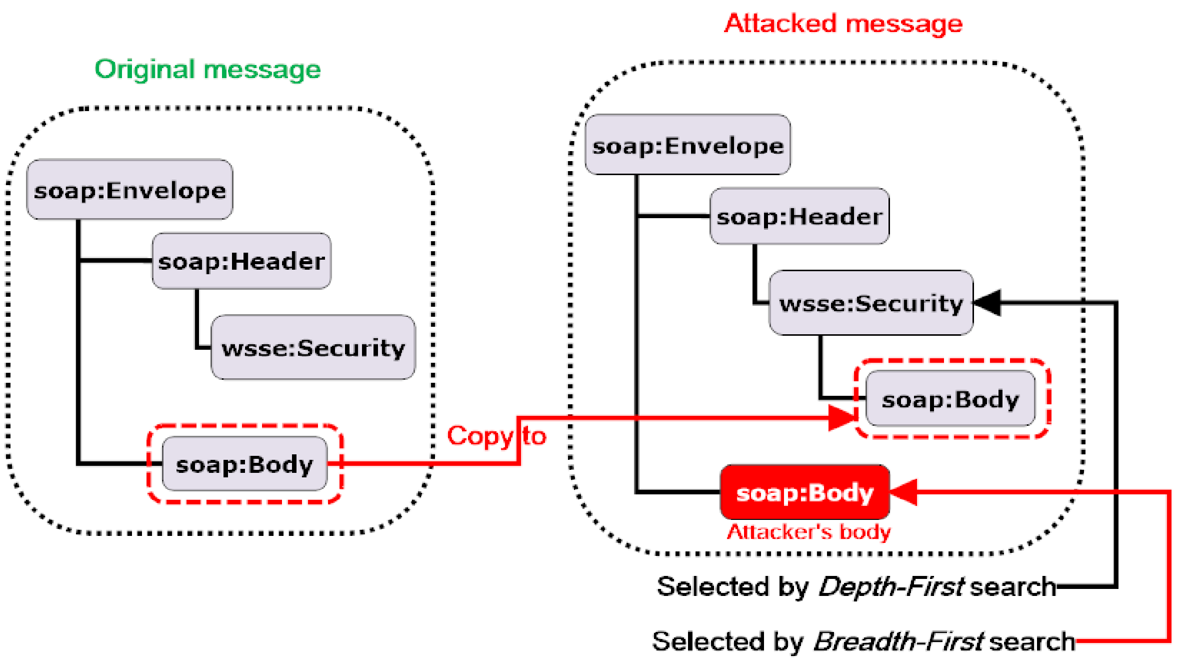


Figure 9.8. XPath descendant axis attack.

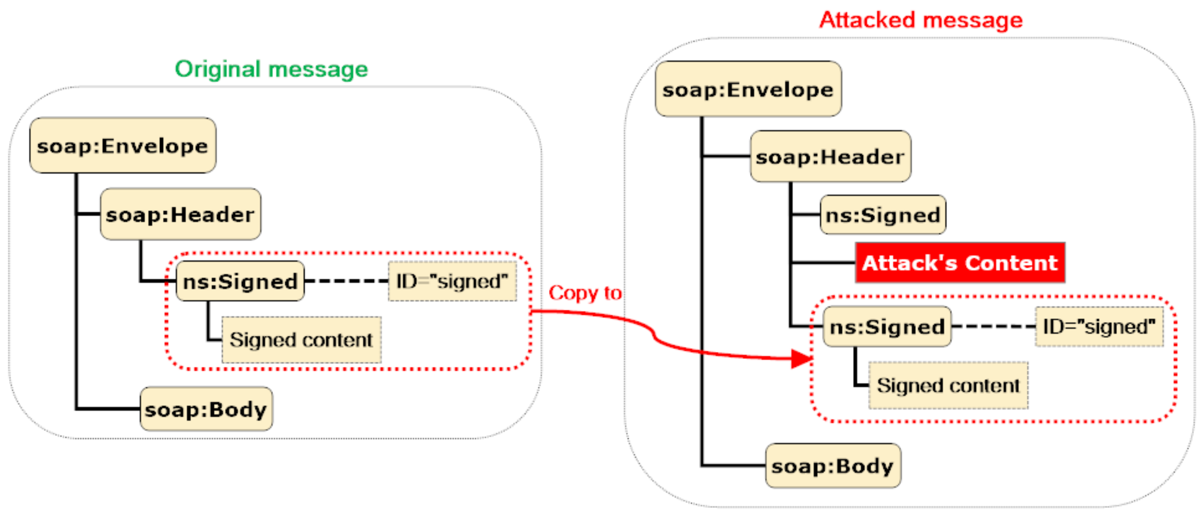


Figure 9.9. XPath attack according to the ID attribute of the signed element.

9.2.2. XML Encryption Attacks and countermeasures

Following are the most important XML encryption attacks including their countermeasures:

1. *Adaptive Chosen-Ciphertext Attack and countermeasure*. In this attack, the attacker tries to decrypt the Ciphertext without knowing the decryption key (no matter if it is symmetric or asymmetric). In example given in figure 9.10, when a sender sends a Ciphertext (*CT*) to a receiver. In the middle, an attacker will receive the CT from the sender and he iteratively generates new Ciphertext (*CT'*, *CT''*, ... etc.) related to the *CT*. The attacker will send these Ciphertext to the receiver, and analysis the receiver's response. Furthermore, the attacker will adapt another new Ciphertext based on this response as well, and repeats this steps until *CT* is decrypted [91][97]. The most important attacks are:
   - The attack on CBC-based symmetric encryption based on side-channel information, which known as called Vaudenay’s attack [98].
   - The attack on RSA-PKCS#1 based on public-key encryption, which known as Bleichenbacher’s attack [99].

   The countermeasure of these attacks can take a place by improving security issues and deploying ciphers secure versus these attacks [97].

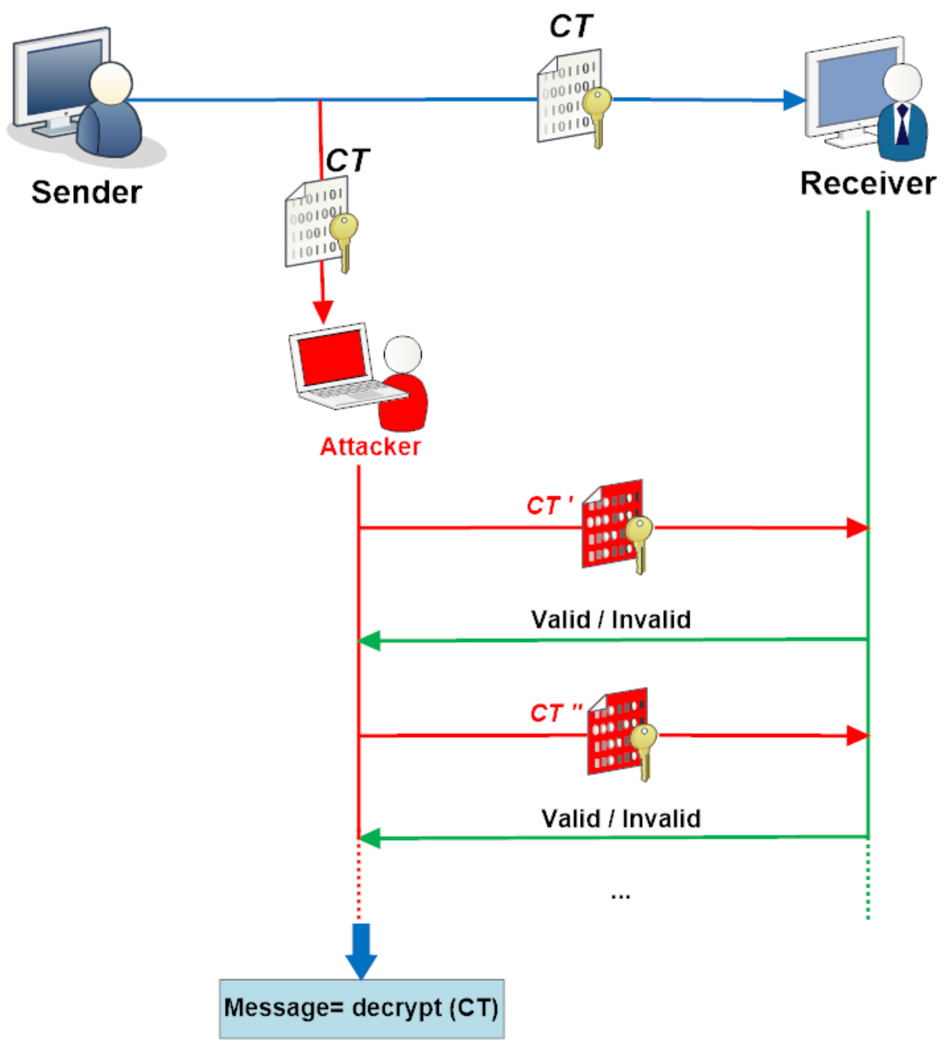


Figure 9.10. Adaptive Chosen-Ciphertext Attack.

2. *XML Encryption Wrapping*. Like XML signature wrapping, the attacker adds a new `<EncryptedData>` element to the header of the SOAP message to force the receiver or the application logic to process it. as well as the attacker keeps the original body element of the SOAP message without change (see figure 9.11). So the web service or a receiver will verify and decrypt the original body element of the SOAP message. As well as, the web service or the receiver will decrypt the new `<EncryptedData>` element (attack's content) which referenced by the attacker's attribute (URI="#oracle") in the `<DataReference>` element [97].

The countermeasure of this attack versus PKCS#1 Ciphertext can be solved by generating random symmetric key whenever the decryption process failed and use the new key for further consideration [99].

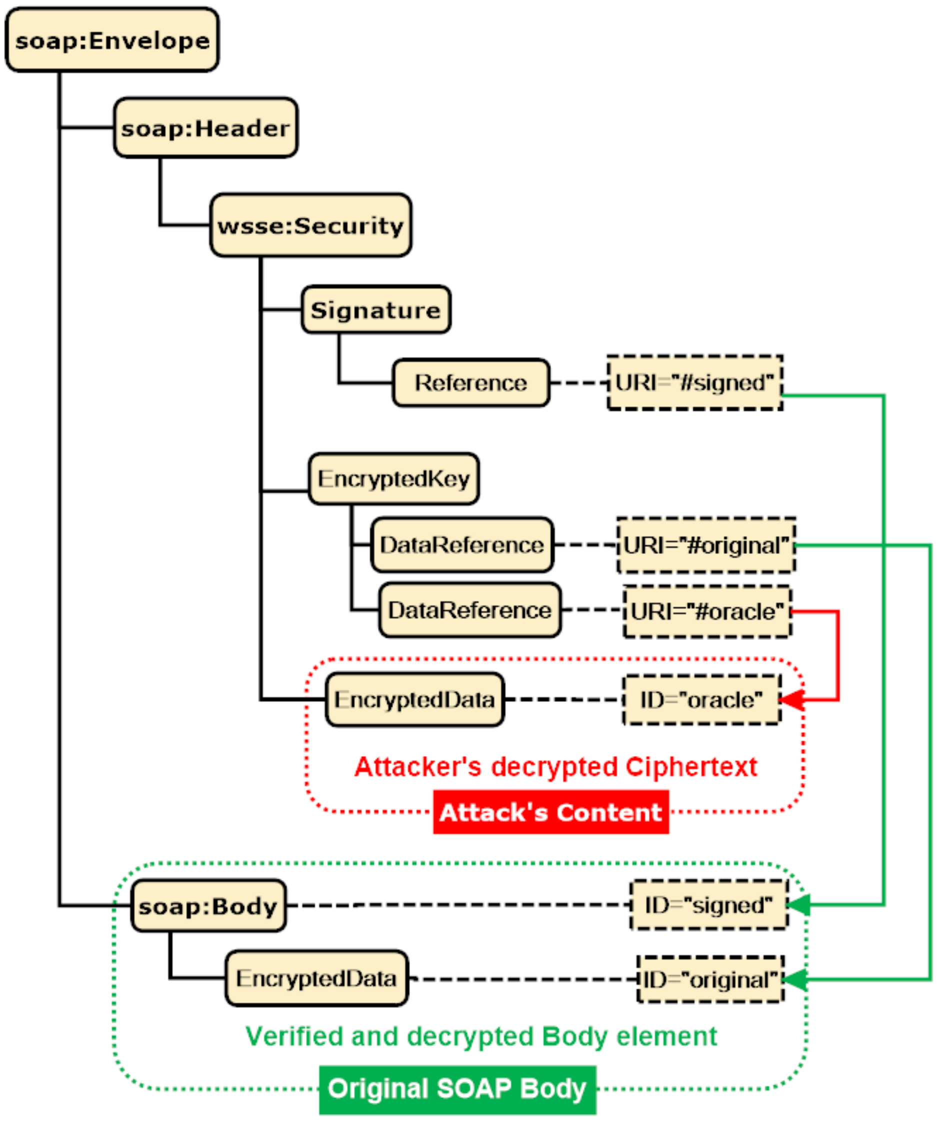


Figure 9.11. Encryption wrapping attack.

10. CONCLUSION

In recent years, the growth of Arabic content and numbers of users on the Internet has greatly increased as can be seen from the table 2.1 and figure 2.1 of top ten languages on the Internet. Arabic is a widely spoken language with more than 375 million speakers and over 155 million, or over forty percent of these Arabic-speaking people use the Internet. This represents nearly five percent of all the Internet users in the world. The number of Arabian speaking Internet users has grown by a factor of sixty in the last fifteen years (2000-2015). This growth in usage has outpaced the growth in information retrieval systems, summarization of Arabic text (such as documents and web pages), query processes and natural language processors.

W3C proposed the XML as standard that used in web applications, transactions, documentations, database management systems and to exchange information between systems over the Internet. XML allows storing different data regardless of how it will be displayed. XML has been used to create, update and query databases. Create and write clear human-readable XML documents as well as machine-readable are easy, so it's easy to create applications that process these XML documents. Generally, all kinds of information can be expressed as XML documents.

This thesis proposed the RAX System which designed for ranking Arabic documents based on content similarity. Our model was applicable to documents stored in different formats and written in the Arabic language. The design and implementation were based on existing text processing frameworks and referent Arabic grammar. The main focus of the research was on evaluating different similarity measures used for classifying Arabic documents from different domains and different categories.

In the preparation stage, the RAX system was used to process Arabic text taking in account the character encoding for the Arabic language (UTF-8, Windows-1256 etc). The preparation stage of the processing of Arabic text was established in 4 steps: extraction of full text from documents; normalization (remove diacritics, remove non-letters and remove punctuation marks); removal of stopwords from the normalized text and stemming (remove prefixes, remove suffixes and finally extract roots or stems words). The well-formed Arabic XML document was created from the stemmed text and loaded into XDBMS which manages end user queries over a collection of XML documents.

In the implementation stage, the RAX system managed XML documents via an XML database management system using XPath and XQuery languages. The Arabic text in queries was processed in 3 steps: normalization, removal of stopwords and stemming (preparation stage). The RAX system uses cosine similarity to measure the similarity metric in n-dimensional space. This is based on the finding that when two vectors are similar in rate and direction from the origin to their end points, they will be close to each other in the vector space, with a small angular separation, and vice versa. The cosine value lies between 1 and -1. Therefore, the cosines of small angles are close to 1, which means high similarity, while the cosines of large angles are close to -1, which means low similarity.

From appendix table 1, appendix table 2 and figure 7.4 it can be seen that where $query_1$ had two terms the result matched 60% of the collection. The top ranked documents which contained both terms were D_{35}, D_{63}, D_{58}, D_{20}, D_{50}, D_{54}, and D_{11}.

From appendix table 3, appendix table 4 and figure 7.5 it can be seen that where $query_2$ had 3 terms there were no documents in the collection which match one of the terms i.e. Computer. In this case, it was impossible to calculate IDF due to the denominator dfj being equal to zero. In this case the RAX system excluded the term Computer from further consideration and $query_2$ became a query of two terms. The $query_2$ result then matched 47% of the collection. The top ranked documents which contained both terms were D_{62}, D_{29}, D_{33}, D_{16}, D_{58}, D_{11}, D_{50}, D_{56}, D_{53}, and D_{36}.

From appendix table 5, appendix table 6 and figure 7.6 it can be seen that where $query_3$ had 3 terms the $query_3$ result matched 79% of the collection. The top ranked documents which contained three terms were D_{20}, D_{51}, D_{56}, D_{59}, D_{87}, D_{23}, D_{31}, D_{38}, D_{12}, D_{48}, D_{66}, D_{24}, D_{55}, D_{63}, D_{10}, and D_{62}.

From appendix table 7, appendix table 8 and figure 7.7 it can be seen that where $query_4$ had 4 terms the $query_4$ result matched 93% of the collection. The top ranked documents which contained four terms were D_{55}, D_{14}, D_{43}, D_{96}, D_{71}, and D_{18}.

We conclude that the Arabic text was fully represented in the processing of Arabic documents.

Furthermore, the total of the term frequencies of the documents and the weights of query$_1$, query$_2$, query$_3$ and query$_4$ were equal to the totals of the whole collection in appendix table 9. There was a proportional relationship between the number of terms of a query and its result. The RAX system excludes terms which are not matched. Some factors such as the position of nodes in the XML tree and the query expressions (structure of expressions) could affect the operation of the RAX system. System performance could be improved by changing the type of stemmer.

There are two main advantages of the RAX system. Firstly, the query results are more comprehensive and wider when using the roots of words or stems. Secondly, the similarity measures are calculated after the completion of the query process i.e. comparing the collection of terms extracted from the collection of XML Arabic documents and the query terms. So, the ranking is calculated according to this comparison.

In section 9, the thesis proposed a survey, which’s studied the security issues of the RAX system. These issues combined between XML security (XML digital signature and XML encryption) and the SOAP message to create a secret environment between an end user and the RAX system model (see figure 9.1) as well as study the security attacks and countermeasures.

Regarding the hypotheses in section 6.2. The verification of these hypotheses as follows:

1. Different forms of a word have caused problems in text processing, document summarization, and information retrieval systems. So, the first hypothesis is true.
2. In every summarization process, there was information loss that directly proportional to the size of the document (the summarized document was less than the original document). So the second hypothesis is true.
3. The well-summarized document contains the whole important information, but with a big document, it’s impossible to get well-summarized document without losing important information. So, the third hypothesis is false.
4. The summarized document always has a smaller size than the original. However, the parsers can process it in the small amount of memory. So, the fourth hypothesis is true, because the summarization process minimizes the consumption of the memory.

5. As we can see from the results of the RAX system that the documents are ordered according to the similarity measures, and this ranking is helpful in information retrieval systems. So, the fifth hypothesis is true.
6. The similarity measures between XML documents and their summarized documents are close. So, the sixth hypothesis is true.
7. The similarity between a query and its result depending on the terms of query and content of summarized XML document. So, the seventh hypothesis is true.
8. If the security attacks and the countermeasures are taking into account. So, the security issues will be powerful and the eighth hypothesis is true.
9. The XML digital signature and the encryption do not affect the summarized XML document. So, the ninth hypothesis is false.

As regards future work, the RAX system could be improved in various ways. We plan to work on making it more efficient. This will mean that the stemmer will need to be improved and enhanced in capabilities and effectiveness to deal with the huge volume of Arabic roots in large data sets (stopword list, compatibility between prefixes and suffixes in stemming process, etc). We also aim to use DTD and XML schema to create XML documents as well as to enhance their summarization. Finally, we plan to upgrade the RAX system to find and replace any query term which has a zero term frequency.

CONFERENCES AND PAPERS

The published conferences and papers are illustrated below:

1. Hesham Elzentani, “An Open-Source Based Application for Creating and Verifying Digital Signatures for XML Documents”, International Conference on Computing, Communication System and Informatics Management (ICCCSIM), Dubai, UAE, 29–30 July, 2012, International Journal of Information Technology and Computer Science (IJITCS), 4(2): 1–10, 2012.

2. Hesham Elzentani, Mladen Veinović, “Summarization of XML Documents”, Konferencije Elektronika, Telekomunikacije, Računarstvo, Automatika i Nuklearna tehnika (ETRAN), Zlatibor, Serbia, pp. VI2.2.1 - 2.2.6, 3–6 June, 2013.

3. Hesham Elzentani, “Managing XML Trees Using XPath, XQuery, Clustering and Tree Tuples over Sedna XML database”, 1st Singidunum University International Scientific Conference SINTEZA, Belgrade, Serbia, ISBN: 978-86-7912-539-2, pp. 878–881, 25–26 April, 2014, DOI: 10.15308/SInteZa-2014-878-881.

4. Hesham Elzentani, Mladen Veinović and Goran Šimić, “Managing Semi-Structured Data Using XPath, XML Trees and Tree Tuples over a Wireless Network”, Special issue of International Journal of Information Technology and Computer Science (IJITCS), 13(1): 45–55, 2014.

The following papers are submitted for publication:

1. Hesham Elzentani, Mladen Veinović and Goran Šimić, “RAX system to rank Arabic XML documents”, Submitted to Kuwait Journal of Science (KJSE, ISSN: 1024-8684) on 21/03/2016.
2. Hesham Elzentani and Mladen Veinović, “Arabic Text: Summarizing and Querying”, Submitted to The International Arab Journal of Information Technology (IAJIT, ISSN: 2309-4524) on 04/05/2016.

LITERATURE

[1] Eric Lease Morgan, “Getting Started with XML: A Manual and Workshop”, 2003.

[2] W3C, “Extensible Markup Language (XML) 1.0”, Second Edition, 2000.

[3] Tutorials Point, “XML Tutorial”, Tutorials Point (I) Pvt. Ltd., 2014.

[4] Erik T. Ray, “Learning XML”, First Edition, 2001.

[5] Benoît Marchal, “XML by Example”, Que, 2000.

[6] Andrea Tagarelli and Sergio Greco, “Toward semantic XML clustering”, in: Proc. SIAM Int. Conf. on Data Mining (SDM), pp. 188–199, 2006.

[7] S. Flesca, F. Furfaro, S. Greco, and E. Zumpano, “Repairs and Consistent Answers for XML Data with Functional Dependencies”, In Proc. Int. XML Database Symposium (XSym), pp. 238–253, 2003.

[8] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt and Jeffrey Naughton , “Relational Databases for Querying XML Documents: Limitations and Opportunities”, Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

[9] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy and Dan Suciu,“XML-QL: A Query Language for XML”, Submission to W3C, 1998, https://www.w3.org/TR/NOTE-xml-ql/.

[10] Steven DeRose, Ron Daniel Jr., Paul Grosso, Eve Maler, Jonathan Marsh and Norman Walsh, “XML Pointer Language (XPointer)”, W3C, 2002, https://www.w3.org/TR/xptr/.

[11] Jonathan Robie, “XQL (XML Query Language)”, 1999, http://www.ibiblio.org/xql/xql-proposal.html.

[12] W3C, “XML Path Language (XPath)”, W3C Recommendation, 2010, https://www.w3.org/TR/xpath20/.

[13] W3C, “XQuery 1.0: An XML Query Language”, W3C Recommendation , 2010, https://www.w3.org/TR/xquery/.

[14] Bergeron, Randy, “XPath-Retrieving Nodes from an XML Document”, SQL Server Magazine, 2000, http://sqlmag.com/xml/xpath151retrieving-nodes-xml-document.

[15] Pierre Geneves, “Course: The XPath Language”, 2016.

[16] Pierre Geneves, “Logics for XML”, PhD thesis, 2006.

[17] Priscilla Walmsley, "XQuery", O’Reilly Media Inc., 2007.

[18] W3C, "XQuery 3.0: An XML Query Language", 2014, http://www.w3.org/TR/xquery-30/.

[19] Ronald Bourret, “XML Database Products”, 2010, retrieved 2016, http://www.rpbourret.com/xml/XMLDatabaseProds.htm.

[20] Ronald Bourret, “XML and Databases”, 2005, retrieved 2016, http://www.rpbourret.com/xml/XMLAndDatabases.htm.

[21] Kareem Darwish and Walid Magdy, “Arabic Information Retrieval”, Foundations and Trends in Information Retrieval, 7(4): 239-342, 2013.

[22] Karin C. Ryding, “A Reference Grammar of Modern Standard Arabic”, Cambridge University Press, 2005.

[23] Aitao Chen and Fredric Gey, “Building an Arabic Stemmer for Information Retrieval”, 11th Text Retrieval Conference, TREC 2002, National Institute of Standards and Technology (NIST), 2003.

[24] Anjali Ganesh Jivani , “A Comparative Study of Stemming Algorithms”, International Journal of Computer Technology and Applications (IJCTA), 2(6): 1930-1938, 2011.

[25] Leah S. Larkey, Lisa Ballesteros and Margaret E. Connell, “Light stemming for Arabic information retrieval”, in Arabic Computational Morphology: Text, Speech and Language Technology, Springer, 38: 221-243, 2007.

[26] Mohammed A. Otair, “Comparative analysis of Arabic stemming algorithms”, International Journal of Managing Information Technology (IJMIT), 5(2): 1-12, 2013.

[27] Majdi Sawalha, Eric Atwell and Mohammad A. M. Abushariah, “SALMA: Standard Arabic Language Morphological Analysis”, 1st International Conference on Communications, Signal Processing, and their Applications, IEEE, 2013.

[28] Dipanjan Das and Andre F.T. Martins, “A Survey on Automatic Text Summarization”, Literature Survey for the Language and Statistics II course at CMU 4: 192-195, 2007.

[29] Sherry and Parteek Bhatia, “A Survey to Automatic Summarization Techniques”, International Journal of Engineering Research and General Science, 3(5):1045-1053, 2015.

[30] Teng Lv and Ping Yan, “A Framework of Summarizing XML Documents with Schemas”, The International Arab Journal of Information Technology, 10(1), 2013.

[31] Jakub Marciniak, “XML schema and data summarization”, 10th International Conference of Artificial Intelligence and Soft Computing, Springer, 2010.

[32] Cong Yu and H. V. Jagadish, “Schema Summarization”, In VLDB, pp:319-330, 2006.

[33] Felix Weigel, “Structural Summaries as a Core Technology for Efficient XML Retrieval”, PhD thesis, 2006.

[34] Zi Lin, Bingsheng He and Byron Choi1, “A quantitative summary of XML structures”, Chapter in Conceptual Modeling - ER 2006, Lecture Notes in Computer Science, Springer, (4215): 228-240, 2006.

[35] Sara Comai, Stefania Marrara and Letizia Tanca, “A synopsis based approach for XML fast approximate querying”, Article in Studies in Fuzziness and Soft Computing, 2006.

[36] Maya Ramanath and Kondreddi Sarath Kumar, “A rank-rewrite framework for summarizing XML documents”, in Proceedings of 2nd International Workshop on Ranking in Databases, ICDE Workshop, Mexico, pp. 540-547, 2008.

[37] Grzegorz Kondrak, “N-gram similarity and distance”, String Processing and Information Retrieval, Lecture Notes in Computer Science, 3772: 115-126, 2005.

[38] Goran Šimić, Zoran Jeremić, Ejub Kajan, Dragan Randjelović and Aaron Presnall, “A framework for delivering e-government support”, Acta Polytechnica Hungarica, 11(1): 79-96, 2014.

[39] Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman, “Mining of massive datasets”, Chapter 3, Cambridge University Press, 2nd edition, 2014.

[40] Hani Abu-Salem, Mahmoud Al-Omari and Martha W. Evens, “Stemming methodologies over individual query words for an Arabic information retrieval system”, Journal of the American Society for Information Science, 50(6):524–529, 1999.

[41] Torsten Shlieder and Holger Meuss, “Querying and ranking XML documents”, Journal of the American Society for Information Science and Technology, 53(6):489-503, 2002.

[42] Dik L. Lee, Huei Chuang and Kent Seamons, “Document ranking and the vector-space model”, IEEE Software, 14(2): 67-75, 1997.

[43] Hahn U. and Mani I., “The Challenges of Automatic Summarization”, Journal of Computer, 33(11): 29-36, 2000.

[44] Amini M., Tombros A., Usunier N., and Lalmas M., "Learning-Based Summarization of XML Documents", Information Retrieval, 10(3): 233-255, 2007.

[45] Yu C. and Jagadish H., "Schema Summarization", in Proceedings of the 32nd International Conference on Very Large Data Bases VLDB, Korea, pp. 319-330, 2006.

[46] Theodore Dalamagas, Tao Cheng, Klaas-Jan Winkel and Timos Sellis, "A Methodology for Clustering XML Documents by Structure", Information Systems, 31(3): 187-228, 2006.

[47] Christopher League and Kenjone Eng, "Schema-Based Compression of XML Data with Relax NG", journal of computers, 2(10): 9-17, 2007.

[48] Sebastian Maneth, Nikolay Mihaylov and Sherif Sakr, "XML Tree Structure Compression", DEXA Workshop: Xantec 2008, pp. 243-247, Turin, Italy, September, 2008.

[49] Juliana Freire, Jayant R. Haritsa, Maya Ramanath, Prasan Roy, and Jerome Simeon, "StatiX: Making XML Count", in Proceedings of the International Conference on Management of Data, USA, pp. 181-191, 2002.

[50] Neoklis Polyzotis, Minos Garofalakis and Yannis Ioannidis, "Approximate XML Query Answers", in Proceedings of SIGMOD International Conference on Management of Data, France, pp. 263-274, 2004.

[51] Wei Wang, Haifeng Jiang, Hongjun Lu and Jeffrey Xu Yu, "Bloom Histogram: Path Selectivity Estimation for XML Data with Updates", in Proceedings of the 30th International Conference on Very Large Data Bases VLDB, Canada, pp. 240-251, 2004.

[52] Ning Zhang, M. Tamer Ozsu, Ashraf Aboulnaga and Ihab F. Ilyas, "XSEED: Accurate and Fast Cardinality Estimation for XPath Queries", in Proceedings of the 22nd International Conference on ICDE, USA, pp. 61, 2006.

[53] Veronica Mayorga and Neoklis Polyzotis, "Sketch-based Summarization of Ordered XML Streams", in Proceedings of IEEE 25th International Conference on ICDE, China, pp. 541-552, 2009.

[54] Gudrun Fischer and Igor Jacy Lino Campista, "A Template-Based Approach to Summarize XML Collections", in Proceedings of Lernen, Wissensentdeckung and Adaptivit , Germany, pp. 103-108, 2005.

[55] Jose de Aguiar and Theo Harder, “EXsum - An XML Summarization Framework”, in Proceeding of 12th International Database Engineering and Applications Symposium (IDEAS 2008), pp. 139-148, 2008, Coimbra, Portugal.

[56] Qinghua Zou, Shaorong Liu and Wesley W. Chu, “Using a compact tree to index and query XML data”, in Proceedings of the 13th ACM International Conference on Information and Knowledge Management, pp. 234-235, 2004.

[57] Lijing Zhang, Xiaoxiao Shen and Wei Xiong, “The query and application of XML data based on XQuery”, 4th International Conference on Computational and Information Sciences, 2012.

[58] Stuart Rose, Dave Engel, Nick Cramer and Wendy Cowley, “ Automatic keyword extraction from individual documents”, in Text Mining: Applications and Theory, edited by Michael W. Berry and Jacob Kogan. 2010.

[59] Moath M. Najeeb, Abdelkarim A. Abdelkader and Musab B. Al-Zghoul, “Arabic natural language processing laboratory serving Islamic sciences”, International Journal of Advanced Computer Science and Applications, 5(3): 114-117, 2014.

[60] Jafar Ababneh, Omar Almomani, Wael Hadi, Nidhal Kamel Taha El-Omari and Ali Al-Ibrahim, “Vector space models to classify Arabic text”, International Journal of Computer Trends and Technology, 7(4): 219-223, 2014.

[61] Spence Green, Conal Sathi and Christopher D. Manning, “NP subject detection in verb-initial Arabic clauses”, Proceedings of the 3rd Workshop on Computational Approaches to Arabic Script-based Languages, 2009.

[62] Barbara Carminati, Elena Ferrari and Elisa Bertino, “A Comprehensive Framework for Secure Outsourcing of XML Data”, Journal of Information Assurance and Security, 3: 289-303, 2008.

[63] Ernesto Damiani, Majirus Fansi, Alban Gabillon and Stefania Marrara, “A General Approach to Securely Querying XML”, Computer Standards & Interfaces, 30(6): 379-389, 2008.

[64] Seifedine Kadry, “Document Security Using XML Technology”, Proceedings of International Conference on Electronic Engineering and Computer Science, Published by Elsevier B.V, pp. 1-6, 2013.

[65] Chi Po Cheong, Chris Chatwin and Rupert Young, “Enhanced and Sustainable WS-Security Using the Participant Doman Name Token”, Journal of Emerging Technologies in Web Intelligence, 6(3): 305-317, 2014.

[66] Andrew Clarke, Eric Pardede and Robert Steele, “External and Distributed Databases: Efficient and Secure XML Query Assurance”, International Journal of Computational Intelligence Systems, 5(3): 421-433, 2012.

[67] M.K.H. Chowdhury, M. Samsuzzaman, T. Islam and B.M. Solaiman, “Proposed Technique of XML Base Secured Data Encryption and Transmission Technology”, World Applied Sciences Journal, 20(7): 941-945, 2012.

[68] Galoh Rashidah Haron, Dharmadharshni Maniam and Shawn Tan Ser Ngiap, “Revisiting Secure Documents with XML Security: A Component-based Approach”, Proceedings of Developments in E-Systems Engineering (DESE), IEEE Computer Society, pp. 257-262, 2010.

[69] P.Dhivya, S. Karthik and T.Kalaikumaran, “SOA based Secure Data Transmission over CMEA Protocol in MANET”, Procedings of Computer Science, Elsevier B.V, 47: 434-440, 2015.

[70] Jonathan Hedley, “jsoup Java HTML Parser”, accessed April 1, 2016, http://jsoup.org.

[71] “Apache POI - Text Extraction”, The Apache Software Foundation, accessed April 1, 2016, https://poi.apache.org/text-extraction.html.

[72] “Apache Tika”, The Apache Software Foundation, accessed April 1, 2016, https://tika.apache.org.

[73] “Apache PDFBox”, The Apache Software Foundation, accessed April 1, 2016, https://pdfbox.apache.org.

[74] “iText®, a Java PDF library”, iText Software, accessed April 1, 2016, https://sourceforge.net/projects/itext.

[75] “Java API for XML Processing (JAXP)”, Oracle Java Documentation, accessed April 1, 2016, https://docs.oracle.com/javase/tutorial/jaxp.

[76] David Megginson, “SAX”, accessed April 1, 2016, http://www.saxproject.org.

[77] Noakes-Fry, Kristen., “Digital Signatures: Perspective”, 2000.

[78] Jonathan Katz, “Digital Signatures”, Springer, 2010.

[79] Blake Dournaee, “XML Security”, McGraw-Hill, 2002.

[80] Ed Simon, Paul Madsen and Carlisle Adams, “An Introduction to XML Digital Signatures”, 2001.

[81] Federal Information Processing Standards Publication 186-4, “Digital Signature Standard (DSS)”, National Institute of Standards and Technology, 2013.

[82] Chafic Maroun, Rouhana Moussa, “Digital Signature and Multiple Signature: Different Cases for Different Purposes”, SANS Institute, 2003.

[83] Yuri Demchenko, “Providing Integrity and Confidentiality with the XML Security: Digital Signature and XML Encryption overview and usage examples”, Draft version 0.1, 2005.

[84] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia and Ed Simon, “XML-Signature Syntax and Processing, W3C Recommendation, Second edition”, 2008, https://www.w3.org/TR/xmldsig-core/.

[85] Takeshi Imamura, Blair Dillaway, Ed Simon, Kelvin Yiu and Magnus Nyström, “XML Encryption Syntax and Processing Version 1.1”, W3C Recommendation, 2013, https://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/#bib-XMLSCHEMA-1.

[86] Takeshi Imamura, Andy Clark and Hiroshi Maruyama, “A Stream-based Implementation of XML Encryption”, ACM Workshop on XML Security, 2002.

[87] Jae-Gil Lee and Kyu-Young Whang, “Secure query processing against encrypted XML data using Query-Aware Decryption”, International Journal of Information Sciences, Elsevier Inc., 176: 1928–1947, 2006.

[88] Sebastian Gajek, Lijun Liao and Jörg Schwenk, “Breaking and fixing the inline approach”, Proceedings of the 2007 ACM workshop on Secure web services, 2007.

[89] W3C, “SOAP Version 1.2 Part 1: Messaging Framework”, Second Edition, 2007.

[90] Michael McIntosh, Paula Austel, “XML Signature Element Wrapping Attacks and Countermeasures”, in SWS ’05: Proceedings of the 2005 workshop on Secure Web Services.

[91] Juraj Somorovsky, “On the insecurity of XML Security”, it – Information Technology, 56(6): 313– 317, 2014, DOI: 10.1515/itit-2014-1045.

[92] Sebastian Gajek, Meiko Jensen, Lijun Liao, and Jörg Schwenk, “Analysis of Signature Wrapping Attacks and Countermeasures”, in the proc of IEEE International Conference on Web services, ICWS, Los Angeles, CA, pp. 575 – 582, 2009.

[93] M. Jensen, C. Meyer, J. Somorovsky, and J. Schwenk. “On the effectiveness of XML schema validation for countering XML signature wrapping attacks”. In Securing Services on the Cloud (IWSSC), 1st International Workshop pp. 7 –13, 2011.

[94] Rahaman M. A. and Schaad A., "Soap-based secure conversation and collaboration. In ICWS, pp. 471–480, 2007.

[95] Romaric Tchokpon and Stelvio Cimato, "Ensuring XML Integrity Using Watermarking Techniques", 8th International Conference on Signal Image Technology and Internet Based Systems, 2012, DOI: 10.1109/SITIS.2012.101.

[96] Lijun Liao, Meiko Jensen, Florian Kohlar, and Nils Gruschka, "On interoperability failures in ws-security: The XML signature wrapping attack", Electronic Business Interoperability: Concepts, Opportunities and Challenges, Information Science Reference, 2011, DOI: 10.4018/978-1-60960-485-1.ch025.

[97] Dennis Kupser, Christian Mainka, Jorg Schwenk and Juraj Somorovsky, "How to Break XML Encryption – Automatically", In Pro-cee-dings of the 9th USENIX Workshop on Offensive Technologies (WOOT), 2015.

[98] Serge Vaudenay, "Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS...", Advances in Cryptology EUROCRYPT'02, Lecture Notes in Computer Science, 2332 : 534–545, Springer-Verlag, 2002.

[99] Daniel Bleichenbacher, "Chosen Ciphertext attacks against protocols based on the RSA encryption standard PKCS #1", Advances in Cryptology – CRYPTO '98, Lecture Notes in Computer Science, 1462: 1–12, Springer, 2006.

APPENDIX

1. Query $_1$ which has the terms { Wire (سلك), Network (شبك)}.

Appendix Table 1. Query$_1$ terms.

Query terms	**TF**	**W_{Qj}**
Wire (سلك)	2.0	1.34488
Network (شبك)	1.0	0.47141

Appendix Table 2. Similarity calculations of query$_1$.

Documents	**Wire** *Documents found= 27 IDF= 0.67244*		**Network** *Documents found=51 IDF= 0.47141*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_1	5	3.3622	0	0	0.94371
D_2	0	0	0	0	Φ
D_3	0	0	3	1.41422	0.33079
D_4	1	0.67244	0	0	0.94371
D_5	0	0	0	0	Φ
D_6	0	0	3	1.41422	0.33079
D_7	2	1.34488	3	1.41422	0.89003
D_8	1	0.67244	2	0.94281	0.81729
D_9	2	1.34488	0	0	0.94371
D_{10}	0	0	5	2.35703	0.33079
D_{11}	24	16.13856	1	0.47141	0.95296
D_{12}	0	0	0	0	Φ
D_{13}	1	0.67244	2	0.94281	0.81729
D_{14}	0	0	11	5.18547	0.33079
D_{15}	3	2.01732	0	0	0.94371
D_{16}	0	0	9	4.24266	0.33079
D_{17}	0	0	0	0	Φ

Documents	**Wire** *Documents found= 27* *IDF= 0.67244*		**Network** *Documents found=51* *IDF= 0.47141*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_{18}	0	0	0	0	Φ
D_{19}	0	0	1	0.47141	0.33079
D_{20}	1	0.67244	1	0.47141	0.96262
D_{21}	0	0	3	1.41422	0.33079
D_{22}	0	0	1	0.47141	0.33079
D_{23}	1	0.67244	6	2.82844	0.54009
D_{24}	1	0.67244	27	12.72798	0.38011
D_{25}	0	0	3	1.41422	0.33079
D_{26}	0	0	1	0.47141	0.33079
D_{27}	0	0	0	0	Φ
D_{28}	0	0	6	2.82844	0.33079
D_{29}	0	0	0	0	Φ
D_{30}	0	0	3	1.41422	0.33079
D_{31}	0	0	26	12.25658	0.33079
D_{32}	0	0	2	0.94281	0.33079
D_{33}	0	0	0	0	Φ
D_{34}	1	0.67244	5	2.35703	0.577
D_{35}	3	2.01732	1	0.47141	0.99422
D_{36}	0	0	11	5.18547	0.33079
D_{37}	0	0	0	0	Φ
D_{38}	1	0.67244	2	0.94281	0.81729
D_{39}	0	0	0	0	Φ
D_{40}	0	0	1	0.47141	0.33079
D_{41}	0	0	9	4.24266	0.33079
D_{42}	0	0	1	0.47141	0.33079
D_{43}	0	0	2	0.94281	0.33079
D_{44}	0	0	1	0.47141	0.33079

Documents	**Wire** *Documents found= 27* *IDF= 0.67244*		**Network** *Documents found=51* *IDF= 0.47141*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_{45}	0	0	0	0	Φ
D_{46}	0	0	0	0	Φ
D_{47}	0	0	0	0	Φ
D_{48}	2	1.34488	5	2.35703	0.75499
D_{49}	1	0.67244	3	1.41422	0.70398
D_{50}	1	0.67244	1	0.47141	0.96262
D_{51}	0	0	3	1.41422	0.33079
D_{52}	0	0	0	0	Φ
D_{53}	2	1.34488	12	5.65688	0.54009
D_{54}	1	0.67244	1	0.47141	0.96262
D_{55}	0	0	2	0.94281	0.33079
D_{56}	0	0	11	5.18547	0.33079
D_{57}	0	0	8	3.77125	0.33079
D_{58}	7	4.70708	1	0.47141	0.97197
D_{59}	0	0	3	1.41422	0.33079
D_{60}	0	0	3	1.41422	0.33079
D_{61}	0	0	0	0	Φ
D_{62}	0	0	0	0	Φ
D_{63}	6	4.03464	5	2.35703	0.9817
D_{64}	0	0	1	0.47141	0.33079
D_{65}	0	0	1	0.47141	0.33079
D_{66}	2	1.34488	0	0	0.94371
D_{67}	0	0	0	0	Φ
D_{68}	0	0	0	0	Φ
D_{69}	0	0	3	1.41422	0.33079
D_{70}	0	0	0	0	Φ
D_{71}	0	0	0	0	Φ

Documents	**Wire** *Documents found= 27* *IDF= 0.67244*		**Network** *Documents found=51* *IDF= 0.47141*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_{72}	0	0	2	0.94281	0.33079
D_{73}	0	0	0	0	Φ
D_{74}	0	0	0	0	Φ
D_{75}	0	0	0	0	Φ
D_{76}	0	0	0	0	Φ
D_{77}	0	0	0	0	Φ
D_{78}	0	0	0	0	Φ
D_{79}	0	0	0	0	Φ
D_{80}	1	0.67244	0	0	0.94371
D_{81}	0	0	0	0	Φ
D_{82}	0	0	2	0.94281	0.33079
D_{83}	2	1.34488	0	0	0.94371
D_{84}	0	0	1	0.47141	0.33079
D_{85}	0	0	0	0	Φ
D_{86}	0	0	0	0	Φ
D_{87}	0	0	2	0.94281	0.33079
D_{88}	0	0	0	0	Φ
D_{89}	1	0.67244	0	0	0.94371
D_{90}	0	0	0	0	Φ
D_{91}	0	0	0	0	Φ
D_{92}	0	0	0	0	Φ
D_{93}	0	0	0	0	Φ
D_{94}	0	0	0	0	Φ
D_{95}	0	0	0	0	Φ
D_{96}	0	0	0	0	Φ
D_{97}	1	0.67244	7	3.29985	0.51256
D_{98}	0	0	0	0	Φ

Documents	**Wire** *Documents found= 27* *IDF= 0.67244*		**Network** *Documents found=51* *IDF= 0.47141*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_{99}	0	0	0	0	Φ
D_{100}	2	1.34488	0	0	0.94371
Total	**76**	**51.10544**	**229**	**107.95213**	

2. Query$_2$ which has the terms {Maintain (صون), Computer (كمبيوتر), Program (برمج)}.

Appendix Table 3. Query$_2$ terms.

Query terms	**TF**	**W_{Qj}**
Maintain (صون)	1	0.61542
Computer (كمبيوتر)	1	-
Program (برمج)	1	0.69897

Appendix Table 4. Similarity calculations of query$_2$.

Documents	**Maintain** *Documents found=32* *IDF= 0.61542*		**Program** *Documents found=25* *IDF= 0.69897*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_1	2	1.23085	0	0	0.66083
D_2	0	0	0	0	Φ
D_3	0	0	0	0	Φ
D_4	0	0	0	0	Φ
D_5	0	0	0	0	Φ
D_6	10	6.15424	0	0	0.66083
D_7	0	0	0	0	Φ
D_8	0	0	0	0	Φ
D_9	4	2.4617	0	0	0.66083
D_{10}	0	0	0	0	Φ
D_{11}	1	0.61542	4	2.79588	0.87505
D_{12}	2	1.23085	0	0	0.66083
D_{13}	0	0	0	0	Φ
D_{14}	0	0	0	0	Φ
D_{15}	1	0.61542	0	0	0.66083
D_{16}	1	0.61542	3	2.09691	0.90626
D_{17}	0	0	0	0	Φ
D_{18}	0	0	0	0	Φ

Documents	**Maintain** *Documents found=32* *IDF= 0.61542*		**Program** *Documents found=25* *IDF= 0.69897*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_{19}	2	1.23085	0	0	0.66083
D_{20}	0	0	0	0	Φ
D_{21}	0	0	5	3.49485	0.75054
D_{22}	0	0	1	0.69897	0.75054
D_{23}	0	0	0	0	Φ
D_{24}	0	0	2	1.39794	0.75054
D_{25}	0	0	0	0	Φ
D_{26}	0	0	0	0	Φ
D_{27}	0	0	0	0	Φ
D_{28}	2	1.23085	0	0	0.66083
D_{29}	2	1.23085	1	0.69897	0.94526
D_{30}	0	0	0	0	Φ
D_{31}	0	0	4	2.79588	0.75054
D_{32}	1	0.61542	0	0	0.66083
D_{33}	4	2.4617	2	1.39794	0.94526
D_{34}	0	0	1	0.69897	0.75054
D_{35}	0	0	1	0.69897	0.75054
D_{36}	1	0.61542	14	9.78558	0.79054
D_{37}	0	0	0	0	Φ
D_{38}	0	0	1	0.69897	0.75054
D_{39}	0	0	0	0	Φ
D_{40}	0	0	0	0	Φ
D_{41}	0	0	0	0	Φ
D_{42}	3	1.84627	0	0	0.66083
D_{43}	0	0	0	0	Φ
D_{44}	0	0	0	0	Φ
D_{45}	0	0	0	0	Φ

Documents	**Maintain** *Documents found=32* *IDF= 0.61542*		**Program** *Documents found=25* *IDF= 0.69897*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_{46}	0	0	0	0	Φ
D_{47}	1	0.61542	0	0	0.66083
D_{48}	0	0	3	2.09691	0.75054
D_{49}	4	2.4617	0	0	0.66083
D_{50}	1	0.61542	5	3.49485	0.85377
D_{51}	0	0	1	0.69897	0.75054
D_{52}	0	0	3	2.09691	0.75054
D_{53}	1	0.61542	7	4.89279	0.82714
D_{54}	0	0	0	0	Φ
D_{55}	1	0.61542	0	0	0.66083
D_{56}	1	0.61542	6	4.19382	0.83853
D_{57}	1	0.61542	0	0	0.66083
D_{58}	3	1.84627	1	0.69897	0.88376
D_{59}	2	1.23085	0	0	0.66083
D_{60}	0	0	0	0	Φ
D_{61}	6	3.69254	0	0	0.66083
D_{62}	2	1.23085	4	2.79588	0.95318
D_{63}	0	0	48	33.55056	0.75054
D_{64}	2	1.23085	0	0	0.66083
D_{65}	0	0	0	0	Φ
D_{66}	0	0	0	0	Φ
D_{67}	0	0	0	0	Φ
D_{68}	0	0	0	0	Φ
D_{69}	0	0	0	0	Φ
D_{70}	0	0	0	0	Φ
D_{71}	1	0.61542	0	0	0.66083
D_{72}	0	0	1	0.69897	0.75054

Documents	**Maintain** *Documents found=32* *IDF= 0.61542*		**Program** *Documents found=25* *IDF= 0.69897*		Cos(q, D_i)
	TF	W_{ij}	TF	W_{ij}	
D_{73}	0	0	0	0	Φ
D_{74}	0	0	0	0	Φ
D_{75}	0	0	2	1.39794	0.75054
D_{76}	0	0	2	1.39794	0.75054
D_{77}	0	0	0	0	Φ
D_{78}	0	0	0	0	Φ
D_{79}	0	0	0	0	Φ
D_{80}	1	0.61542	0	0	0.66083
D_{81}	0	0	0	0	Φ
D_{82}	0	0	6	4.19382	0.75054
D_{83}	0	0	0	0	Φ
D_{84}	0	0	0	0	Φ
D_{85}	4	2.4617	0	0	0.66083
D_{86}	0	0	0	0	Φ
D_{87}	0	0	0	0	Φ
D_{88}	3	1.84627	0	0	0.66083
D_{89}	0	0	0	0	Φ
D_{90}	0	0	0	0	Φ
D_{91}	0	0	0	0	Φ
D_{92}	0	0	0	0	Φ
D_{93}	0	0	0	0	Φ
D_{94}	0	0	0	0	Φ
D_{95}	0	0	0	0	Φ
D_{96}	0	0	0	0	Φ
D_{97}	1	0.61542	0	0	0.66083
D_{98}	1	0.61542	0	0	0.66083
D_{99}	0	0	0	0	Φ

Documents	**Maintain** *Documents found=32* *IDF= 0.61542*		**Program** *Documents found=25* *IDF= 0.69897*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	
D_{100}	0	0	0	0	Φ
Total	**72**	**44.31049**	**128**	**89.46816**	

3. **Query$_3$** which has the terms {Language (لغا), Arab (عرب), Home (وطن)}.

Appendix Table 5. Query$_3$ terms.

Query terms	**TF**	**W$_{Qj}$**
Language (لغا)	1	0.48299
Arab (عرب)	2	0.88005
Home (وطن)	1	0.44494

Appendix Table 6. Similarity calculations of query$_3$.

Documents	**Language** *Documents found=49 IDF= 0.48299*		**Arab** *Documents found=57 IDF= 0.44002*		**Home** *Documents found= 56 IDF=0.44494*		**Cos(Q, D$_i$)**
	TF	**W$_{ij}$**	**TF**	**W$_{ij}$**	**TF**	**W$_{ij}$**	
D$_1$	3	1.44897	1	0.44002	1	0.44494	0.74148
D$_2$	1	0.48299	31	13.64077	0	0	0.81652
D$_3$	3	1.44897	1	0.44002	0	0	0.65376
D$_4$	0	0	6	2.64015	0	0	0.80146
D$_5$	0	0	0	0	0	0	Φ
D$_6$	1	0.48299	0	0	0	0	0.43986
D$_7$	2	0.96598	3	1.32007	37	16.46265	0.49285
D$_8$	1	0.48299	68	29.92169	15	6.67405	0.87727
D$_9$	9	4.34691	28	12.32069	33	14.68291	0.90241
D$_{10}$	6	2.89794	6	2.64015	8	3.55949	0.91272
D$_{11}$	271	130.8903	128	56.32317	4	1.77975	0.72583
D$_{12}$	3	1.44897	12	5.2803	2	0.88987	0.94277
D$_{13}$	1	0.48299	68	29.92169	15	6.67405	0.87727
D$_{14}$	0	0	76	33.44188	5	2.22468	0.82659
D$_{15}$	2	0.96598	6	2.64015	7	3.11456	0.90638
D$_{16}$	4	1.93196	3	1.32007	1	0.44494	0.87667
D$_{17}$	1	0.48299	24	10.5606	0	0	0.82072
D$_{18}$	0	0	0	0	1	0.44494	0.4052

Documents	**Language** *Documents found=49 IDF= 0.48299*		**Arab** *Documents found=57 IDF= 0.44002*		**Home** *Documents found= 56 IDF=0.44494*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{19}	0	0	0	0	1	0.44494	0.4052
D_{20}	2	0.96598	3	1.32007	2	0.88987	0.98996
D_{21}	4	1.93196	0	0	0	0	0.43986
D_{22}	0	0	1	0.44002	0	0	0.80146
D_{23}	6	2.89794	22	9.68055	5	2.22468	0.96015
D_{24}	8	3.86392	30	13.20074	3	1.33481	0.92771
D_{25}	1	0.48299	0	0	2	0.88987	0.56595
D_{26}	0	0	2	0.88005	0	0	0.80146
D_{27}	1	0.48299	0	0	0	0	0.43986
D_{28}	0	0	8	3.5202	0	0	0.80146
D_{29}	0	0	140	61.60347	41	18.2424	0.88353
D_{30}	0	0	6	2.64015	4	1.77975	0.89106
D_{31}	4	1.93196	6	2.64015	1	0.44494	0.95287
D_{32}	0	0	4	1.7601	3	1.33481	0.88344
D_{33}	2	0.96598	0	0	2	0.88987	0.59805
D_{34}	1	0.48299	29	12.76072	2	0.88987	0.84371
D_{35}	7	3.38093	0	0	2	0.88987	0.52851
D_{36}	2	0.96598	3	1.32007	0	0	0.90654
D_{37}	1	0.48299	0	0	0	0	0.43986
D_{38}	7	3.38093	9	3.96022	2	0.88987	0.9506
D_{39}	0	0	1	0.44002	1	0.44494	0.85167
D_{40}	8	3.86392	134	58.96332	15	6.67405	0.86875
D_{41}	4	1.93196	0	0	0	0	0.43986
D_{42}	0	0	26	11.44064	10	4.44937	0.89383
D_{43}	0	0	2	0.88005	3	1.33481	0.77945
D_{44}	0	0	29	12.76072	5	2.22468	0.85914

Documents	**Language** *Documents found=49 IDF= 0.48299*		**Arab** *Documents found=57 IDF= 0.44002*		**Home** *Documents found= 56 IDF=0.44494*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{45}	8	3.86392	0	0	0	0	0.43986
D_{46}	0	0	0	0	1	0.44494	0.4052
D_{47}	7	3.38093	0	0	1	0.44494	0.48897
D_{48}	1	0.48299	7	3.08017	5	2.22468	0.93535
D_{49}	0	0	16	7.0404	5	2.22468	0.8863
D_{50}	6	2.89794	0	0	1	0.44494	0.49626
D_{51}	2	0.96598	3	1.32007	1	0.44494	0.98111
D_{52}	1	0.48299	0	0	0	0	0.43986
D_{53}	1	0.48299	0	0	0	0	0.43986
D_{54}	1	0.48299	0	0	1	0.44494	0.59805
D_{55}	1	0.48299	8	3.5202	2	0.88987	0.92667
D_{56}	2	0.96598	4	1.7601	1	0.44494	0.98024
D_{57}	0	0	11	4.84027	5	2.22468	0.89744
D_{58}	2	0.96598	0	0	12	5.33924	0.47704
D_{59}	6	2.89794	22	9.68055	14	6.22911	0.9736
D_{60}	0	0	0	0	0	0	Φ
D_{61}	2	0.96598	1	0.44002	3	1.33481	0.77308
D_{62}	2	0.96598	10	4.40025	1	0.44494	0.91271
D_{63}	4	1.93196	18	7.92045	2	0.88987	0.92162
D_{64}	0	0	0	0	1	0.44494	0.4052
D_{65}	8	3.86392	134	58.96332	15	6.67405	0.86875
D_{66}	7	3.38093	7	3.08017	8	3.55949	0.93142
D_{67}	0	0	0	0	0	0	Φ
D_{68}	0	0	0	0	1	0.44494	0.4052
D_{69}	0	0	9	3.96022	0	0	0.80146
D_{70}	0	0	0	0	0	0	Φ

Documents	**Language** *Documents found=49 IDF= 0.48299*		**Arab** *Documents found=57 IDF= 0.44002*		**Home** *Documents found= 56 IDF=0.44494*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{71}	0	0	1	0.44002	0	0	0.80146
D_{72}	0	0	0	0	0	0	Φ
D_{73}	0	0	0	0	0	0	Φ
D_{74}	0	0	0	0	0	0	Φ
D_{75}	0	0	0	0	0	0	Φ
D_{76}	0	0	0	0	0	0	Φ
D_{77}	0	0	0	0	0	0	Φ
D_{78}	0	0	3	1.32007	1	0.44494	0.8889
D_{79}	0	0	6	2.64015	2	0.88987	0.8889
D_{80}	0	0	0	0	1	0.44494	0.4052
D_{81}	0	0	0	0	0	0	Φ
D_{82}	0	0	0	0	0	0	Φ
D_{83}	0	0	1	0.44002	0	0	0.80146
D_{84}	2	0.96598	1	0.44002	0	0	0.73252
D_{85}	0	0	0	0	0	0	Φ
D_{86}	0	0	0	0	0	0	Φ
D_{87}	1	0.48299	5	2.20012	3	1.33481	0.96116
D_{88}	1	0.48299	22	9.68055	3	1.33481	0.86997
D_{89}	0	0	0	0	0	0	Φ
D_{90}	0	0	0	0	0	0	Φ
D_{91}	0	0	1	0.44002	0	0	0.80146
D_{92}	0	0	0	0	0	0	Φ
D_{93}	0	0	0	0	0	0	Φ
D_{94}	0	0	0	0	0	0	Φ
D_{95}	0	0	0	0	0	0	Φ
D_{96}	0	0	9	3.96022	6	2.66962	0.89106

Documents	**Language** *Documents found=49 IDF= 0.48299*		**Arab** *Documents found=57 IDF= 0.44002*		**Home** *Documents found= 56 IDF=0.44494*		**Cos(Q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{97}	0	0	4	1.7601	0	0	0.80146
D_{98}	0	0	0	0	1	0.44494	0.4052
D_{99}	0	0	0	0	0	0	Φ
D_{100}	0	0	4	1.7601	0	0	0.80146
Total	**431**	**208.1687**	**1223**	**538.15026**	**330**	**146.8291**	

4. **Query$_4$** which has the terms {Use (خدم), Develop (نمي), Wind (روح), Sustain (دوم)}.

Appendix Table 7. Query$_4$ terms.

Query terms	**TF**	**W_{Qj}**
Use (خدم)	1	0.32199
Develop (نمي)	1	0.34626
Wind (روح)	1	0.46042
Sustain (دوم)	1	0.50160

Appendix Table 8. Similarity calculations of query$_4$.

Documents	**Use** *Documents found=91 IDF= 0.32199*		**Develop** *Documents found=82 IDF= 0.34626*		**Wind** *Documents found=53 IDF= 0. 46042*		**Sustain** *Documents found=46 IDF= 0.5016*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_1	9	2.89793	6	2.07755	0	0	3	1.50479	0.75035
D_2	27	8.69378	49	16.96662	2	0.92083	0	0	0.57504
D_3	16	5.15187	5	1.73129	1	0.46042	0	0	0.54635
D_4	4	1.28797	2	0.69252	0	0	0	0	0.53993
D_5	0	0	0	0	0	0	0	0	Φ
D_6	25	8.0498	3	1.03877	2	0.92083	2	1.00319	0.56856
D_7	23	7.40582	9	3.11632	3	1.38125	3	1.50479	0.70638
D_8	3	0.96598	8	2.77006	0	0	1	0.5016	0.61682
D_9	11	3.54191	6	2.07755	1	0.46042	3	1.50479	0.77542
D_{10}	4	1.28797	3	1.03877	1	0.46042	0	0	0.69281
D_{11}	16	5.15187	27	9.34895	2	0.92083	1	0.5016	0.62664
D_{12}	1	0.32199	0	0	1	0.46042	0	0	0.67777
D_{13}	3	0.96598	8	2.77006	0	0	1	0.5016	0.61682
D_{14}	5	1.60996	6	2.07755	7	3.22291	1	0.5016	0.85625
D_{15}	34	10.94773	19	6.57889	2	0.92083	2	1.00319	0.63209
D_{16}	33	10.62574	10	3.46258	0	0	3	1.50479	0.57503
D_{17}	2	0.64398	8	2.77006	0	0	4	2.00638	0.75314

Documents	**Use** *Documents found=91 IDF= 0.32199*		**Develop** *Documents found=82 IDF= 0.34626*		**Wind** *Documents found=53 IDF= 0. 46042*		**Sustain** *Documents found=46 IDF= 0.5016*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{18}	25	8.0498	29	10.04147	1	0.46042	14	7.02233	0.80625
D_{19}	4	1.28797	1	0.34626	2	0.92083	0	0	0.7135
D_{20}	28	9.01578	0	0	0	0	0	0	0.38843
D_{21}	20	6.43984	8	2.77006	1	0.46042	2	1.00319	0.6371
D_{22}	30	9.65976	1	0.34626	0	0	0	0	0.40315
D_{23}	2	0.64398	13	4.50135	2	0.92083	0	0	0.56943
D_{24}	41	13.20167	10	3.46258	9	4.14374	1	0.5016	0.64316
D_{25}	40	12.87968	24	8.31018	0	0	5	2.50798	0.64331
D_{26}	29	9.33777	10	3.46258	1	0.46042	1	0.5016	0.56427
D_{27}	0	0	0	0	0	0	0	0	Φ
D_{28}	29	9.33777	7	2.4238	0	0	0	0	0.48092
D_{29}	30	9.65976	27	9.34895	0	0	1	0.5016	0.59178
D_{30}	16	5.15187	308	106.6473	5	2.30208	3	1.50479	0.45632
D_{31}	47	15.13362	3	1.03877	3	1.38125	1	0.5016	0.48444
D_{32}	12	3.8639	15	5.19386	0	0	3	1.50479	0.68927
D_{33}	2	0.64398	3	1.03877	1	0.46042	0	0	0.71956
D_{34}	34	10.94773	3	1.03877	0	0	0	0	0.42615
D_{35}	62	19.9635	25	8.65644	4	1.84166	3	1.50479	0.60778
D_{36}	49	15.77761	8	2.77006	0	0	5	2.50798	0.54294
D_{37}	0	0	0	0	0	0	0	0	Φ
D_{38}	128	41.21497	5	1.73129	2	0.92083	0	0	0.41792
D_{39}	3	0.96598	20	6.92515	1	0.46042	0	0	0.50285
D_{40}	11	3.54191	12	4.15509	0	0	2	1.00319	0.66984
D_{41}	60	19.31952	11	3.80883	1	0.46042	0	0	0.47475
D_{42}	26	8.37179	25	8.65644	1	0.46042	3	1.50479	0.66152
D_{43}	16	5.15187	17	5.88638	11	5.06457	2	1.00319	0.84073

Documents	**Use** *Documents found=91 IDF= 0.32199*		**Develop** *Documents found=82 IDF= 0.34626*		**Wind** *Documents found=53 IDF= 0. 46042*		**Sustain** *Documents found=46 IDF= 0.5016*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{44}	21	6.76183	7	2.4238	0	0	8	4.01276	0.73737
D_{45}	1	0.32199	0	0	0	0	0	0	0.38843
D_{46}	6	1.93195	0	0	11	5.06457	2	1.00319	0.75653
D_{47}	9	2.89793	16	5.54012	0	0	0	0	0.55017
D_{48}	175	56.3486	27	9.34895	0	0	0	0	0.45157
D_{49}	30	9.65976	39	13.50404	1	0.46042	0	0	0.5809
D_{50}	32	10.30374	8	2.77006	0	0	0	0	0.48356
D_{51}	7	2.25394	0	0	0	0	0	0	0.38843
D_{52}	1	0.32199	0	0	0	0	0	0	0.38843
D_{53}	48	15.45561	6	2.07755	2	0.92083	1	0.5016	0.49177
D_{54}	15	4.82988	0	0	1	0.46042	1	0.5016	0.49929
D_{55}	20	6.43984	19	6.57889	14	6.44582	4	2.00638	0.87979
D_{56}	28	9.01578	10	3.46258	0	0	0	0	0.51237
D_{57}	42	13.52366	9	3.11632	0	0	0	0	0.47231
D_{58}	21	6.76183	12	4.15509	4	1.84166	1	0.5016	0.69689
D_{59}	30	9.65976	74	25.62306	1	0.46042	16	8.02552	0.68563
D_{60}	14	4.50789	0	0	1	0.46042	1	0.5016	0.50675
D_{61}	27	8.69378	1	0.34626	3	1.38125	0	0	0.48683
D_{62}	53	17.06557	4	1.38503	0	0	0	0	0.42095
D_{63}	92	29.62326	2	0.69252	2	0.92083	1	0.5016	0.42533
D_{64}	4	1.28797	1	0.34626	2	0.92083	0	0	0.7135
D_{65}	11	3.54191	12	4.15509	0	0	2	1.00319	0.66984
D_{66}	21	6.76183	12	4.15509	1	0.46042	1	0.5016	0.61783
D_{67}	2	0.64398	1	0.34626	0	0	0	0	0.53993
D_{68}	34	10.94773	4	1.38503	1	0.46042	0	0	0.46056
D_{69}	3	0.96598	16	5.54012	0	0	0	0	0.47822

Documents	**Use** *Documents found=91 IDF= 0.32199*		**Develop** *Documents found=82 IDF= 0.34626*		**Wind** *Documents found=53 IDF=0. 46042*		**Sustain** *Documents found=46 IDF=0.5016*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{70}	0	0	4	1.38503	0	0	0	0	0.41771
D_{71}	25	8.0498	10	3.46258	3	1.38125	9	4.51436	0.81095
D_{72}	2	0.64398	0	0	0	0	0	0	0.38843
D_{73}	7	2.25394	0	0	1	0.46042	0	0	0.49174
D_{74}	0	0	46	15.92785	1	0.46042	0	0	0.43358
D_{75}	14	4.50789	3	1.03877	3	1.38125	0	0	0.61148
D_{76}	14	4.50789	3	1.03877	3	1.38125	0	0	0.61148
D_{77}	10	3.21992	5	1.73129	1	0.46042	0	0	0.6051
D_{78}	4	1.28797	25	8.65644	3	1.38125	0	0	0.55116
D_{79}	6	1.93195	30	10.38773	16	7.36665	1	0.5016	0.7358
D_{80}	4	1.28797	9	3.11632	0	0	0	0	0.5344
D_{81}	1	0.32199	9	3.11632	0	0	0	0	0.45542
D_{82}	5	1.60996	0	0	0	0	0	0	0.38843
D_{83}	6	1.93195	25	8.65644	0	0	0	0	0.49229
D_{84}	2	0.64398	2	0.69252	0	0	1	0.5016	0.78744
D_{85}	0	0	0	0	0	0	0	0	Φ
D_{86}	3	0.96598	5	1.73129	1	0.46042	0	0	0.66531
D_{87}	3	0.96598	5	1.73129	0	0	1	0.5016	0.68553
D_{88}	7	2.25394	11	3.80883	0	0	0	0	0.5573
D_{89}	7	2.25394	25	8.65644	0	0	1	0.5016	0.5352
D_{90}	0	0	0	0	0	0	0	0	Φ
D_{91}	9	2.89793	7	2.4238	0	0	1	0.5016	0.64066
D_{92}	3	0.96598	7	2.4238	0	0	1	0.5016	0.6365
D_{93}	7	2.25394	2	0.69252	1	0.46042	0	0	0.59127
D_{94}	0	0	0	0	0	0	0	0	Φ
D_{95}	3	0.96598	3	1.03877	4	1.84166	0	0	0.78809

Documents	**Use** *Documents found=91 IDF= 0.32199*		**Develop** *Documents found=82 IDF= 0.34626*		**Wind** *Documents found=53 IDF= 0. 46042*		**Sustain** *Documents found=46 IDF= 0.5016*		**Cos(q, D_i)**
	TF	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	**TF**	**W_{ij}**	
D_{96}	15	4.82988	14	4.84761	7	3.22291	2	1.00319	0.82541
D_{97}	39	12.55769	5	1.73129	79	36.37283	1	0.5016	0.67772
D_{98}	9	2.89793	27	9.34895	2	0.92083	1	0.5016	0.59385
D_{99}	0	0	0	0	0	0	0	0	Φ
D_{100}	39	12.55769	24	8.31018	2	0.92083	0	0	0.58731
Total	**1951**	**628.20635**	**1340**	**463.98513**	**238**	**109.57901**	**127**	**63.70273**	

5. Collection Data Terms.

Following table contains the totals of term frequencies, inverse documents frequencies and the weights of all terms in the collection.

Appendix Table 9.Total terms in the collection.

S.n	Term	Document Frequency	TF	IDF	$W_{i.j}$
1.	سلك	27	76	0.67244	51.10544
2.	شبك	51	229	0.47141	107.95289
3.	صون	32	72	0.61542	44.31024
4.	برمج	25	128	0.69897	89.46816
5.	لغا	49	431	0.48299	208.16869
6.	عرب	57	1223	0.44002	538.14446
7.	وطن	56	330	0.44494	146.8302
8.	خدم	91	1951	0.32199	628.20249
9.	روح	53	238	0.46042	109.57996
10.	نمي	82	1340	0.34626	463.9884
11.	دوم	46	127	0.5016	63.7032