

METADATA EXTRACTION FROM TEXT IN SOCCER DOMAIN

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZKAN GÖKTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2008

Approval of the thesis

**“METADATA EXTRACTION FROM TEXT IN SOCCER
DOMAIN”**

submitted by **Özkan Göktürk** in partial fulfillment of the requirements for
the degree of **Master of Science in Computer Engineering Department,**
Middle East Technical University by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. Volkan Atalay _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Nihan Kesim Çiçekli _____
Supervisor, **Computer Engineering, METU**

Assoc. Prof. Dr. İlyas Çiçekli _____
Co-supervisor, **Computer Engineering, Bilkent Univ.**

Examining Committee Members:

Prof. Dr. Özgür Ulusoy _____
Computer Engineering, Bilkent Univ.

Assoc. Prof. Dr. Nihan Kesim Çiçekli _____
Computer Engineering, METU

Assoc. Prof. Dr. Ferda Nur Alpaslan _____
Computer Engineering, METU

Assoc. Prof. Dr. Ahmet Coşar _____
Computer Engineering, METU

Assist. Prof. Dr. Tuğba Taşkaya Temizel _____
Information Systems, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Özkan Göktürk

Signature :

ABSTRACT

METADATA EXTRACTION FROM TEXT IN SOCCER DOMAIN

Göktürk, Özkan

M.Sc., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Nihan Kesim Çiçekli

Co-Supervisor: Assoc. Prof. Dr. İlyas Çiçekli

September 2008, 69 pages

Video databases and content based retrieval in these databases have become popular with the improvements in technology. Metadata extraction techniques are used for providing data to video content. One popular metadata extraction technique for multimedia is information extraction from text. For some domains, it is possible to find accompanying text with the video, such as soccer domain, movie domain and news domain. In this thesis, we present an approach of metadata extraction from match reports for soccer domain. The UEFA Cup and UEFA Champions League Match Reports are downloaded from the web site of UEFA by a web-crawler. These match reports are preprocessed by using regular expressions and then important events are extracted by using hand-written rules. In addition to hand-written rules, two different machine learning techniques are applied on match corpus to learn event patterns and automatically extract match events. Extracted events are saved in an MPEG-7 file. A user interface is implemented to query the events in the MPEG-7 match corpus and view the corresponding video segments.

Keywords: Semantic querying of video content, MPEG-7, Information extraction, Video annotation

ÖZ

FUTBOL MAÇ ANLATIMLARINDAN ÜSTVERİ ÇIKARMA

Göktürk, Özkan

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü Bölümü

Tez Yöneticisi: Doç. Dr. Nihan Kesim Çiçekli

Ortak Tez Yöneticisi: Doç. Dr. İlyas Çiçekli

Eylül 2008, 69 sayfa

Video veritabanları ve bu veritabanlarından içerik bilgisinin elde edilmesi teknolojinin gelişmesiyle birlikte popüler olmaya başlamıştır. Video içeriğine yardımcı veriler, bilgi çıkarım teknikleri ile elde edilebilmektedir. Bir popüler yardımcı veri çıkarım tekniği, metinden bilgi çıkarımıdır. Futbol, film ve haber bültenleri gibi alanlar için video ile birlikte ilgili metin bulunabilir. Biz bu tezde futbol alanının maç raporlarından yardımcı veri çıkarımı yaklaşımını ortaya koyuyoruz. UEFA kupası ve UEFA şampiyonlar ligi maç raporları web crawler aracılığı ile UEFA'nın web sitesinden indirilmektedir. Düzenli ifadeler kullanarak, maç raporları işaretlenmekte ve daha sonra olayları bu açıklanan maç raporlarından el ile yazılan kurallar yardımı ile seçilip çıkartılmaktadır. Bunların yanında, iki tane bilgi çıkarımı algoritmasını aynı maç verilerine uygulanılmıştır. Seçip çıkarılan olaylar MPEG-7 dosyasında saklanmaktadır. MPEG-7 maç kısmında sorgulanan olayları kullanan bir arabirim ile ortaya koyuyoruz ve bağlantılı bir maç videosu bulunabilirse, bulunan olaylara denk gelen video parçaları oynatılır.

Anahtar Kelimeler: Anlamsal Video İçerik Sorgulama, MPEG-7, Bilgi Çıkarımı, Video İşaretleme

ACKNOWLEDGMENTS

I would like to thank Assoc. Prof. Dr. Nihan Kesim Çiçekli and Assoc. Prof. Dr. İlyas Çiçekli for their continuous help, guidance and encouragement throughout this study.

I am thankful to my company Siemens EC. supporting my graduate studies.

I would also like to thank Tübitak, since this thesis is a part of the project with reference number 107E234 supported by Tübitak.

Finally I am grateful to my family for their continuous encouragement.

To my dear sister...

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	v
ACKNOWLEDGMENTS	vi
DEDICATON	vii
TABLE OF CONTENTS	viii
LIST OF FIGURES	x
LIST OF TABLES	xii
CHAPTER	
1 INTRODUCTION	1
1.1 Motivation	2
1.2 Contributions of the Thesis	2
1.3 Organization of the Thesis	3
2 RELATED WORK	4
2.1 Information Extraction	4
2.1.1 Information extraction from text	4
2.1.2 Information extraction in soccer domain	5
2.2 Data Models for Multimedia Databases	7
2.2.1 XML Databases	7
2.2.2 MPEG-7	14
3 METADATA EXTRACTION FROM TEXT	21
3.1 Web-Crawler	22
3.2 Text Annotation	24
3.3 Match Event Extraction	27
3.4 MPEG-7 Generation from Match Events	28

4	SUPERVISED LEARNING FOR INFORMATION EXTRACTION	32
4.1	Whisk Algorithm	32
4.1.1	Creating hand-tagged training instances	33
4.1.2	Creating a rule from a seed instance	34
4.2	Event Extraction with Unique Match Sequence	35
4.3	Discussion	37
5	IMPLEMENTATION	39
5.1	Architecture	40
5.2	System Overview	42
5.3	Tools Used for Implementation	45
5.3.1	NEKOHtm1	45
5.3.2	XQEngine	47
5.3.3	Java Media Framework	47
6	CONCLUSION	50
	REFERENCES	52
A	RULE SET FOR MATCH EVENTS	55
B	EXTRACTED RULES FOR EVENT EXTRACTION WITH UNIQUE MATCH SEQUENCE	66

LIST OF FIGURES

FIGURES

Figure 2.1 XML Fragment	7
Figure 2.2 XML DTD	8
Figure 2.3 XML XDR	8
Figure 2.4 XML XSD	9
Figure 2.5 XML Document as a Column of a Table	10
Figure 2.6 XPath Example	11
Figure 2.7 XQuery Path Expression	12
Figure 2.8 XQuery Element Constructor	12
Figure 2.9 XQuery FLWOR Expressions	13
Figure 2.10 XQuery Conditional Expressions	13
Figure 2.11 XQuery Quantified Expressions	14
Figure 2.12 XQuery User Defined Functions	14
Figure 2.13 MPEG-7 area of application. The figure is based on the MPEG-7 standard [5]	17
Figure 2.14 Overview of the MPEG-7 visual features. The figure is based on the MPEG-7 standard [5]	19
Figure 2.15 Semantic Description Example	20
Figure 2.16 Semantic Description Example	20
Figure 3.1 System Overview	22
Figure 3.2 Fixtures and Results of UEFA Champions League	23
Figure 3.3 Qualifying Round of UEFA Champions League	24
Figure 3.4 Example Minute by Minute Match Report	25
Figure 3.5 Regular Expressions	26

Figure 3.6	A Tagged Sentence	26
Figure 3.7	Tagged Sentence with a two tokened player name	26
Figure 3.8	A Sample Rule for Extraction	28
Figure 3.9	A Sample Event Representation	28
Figure 3.10	Example MPEG-7 Descriptors	30
Figure 3.11	Example Foul Event MPEG-7 Descriptors	30
Figure 3.12	Example Foul Committed Event MPEG-7 Descriptors	31
Figure 3.13	Example Foul Suffered Event MPEG-7 Descriptors	31
Figure 4.1	Evaluating Performance of WHISK	33
Figure 4.2	An example for tagged instance for WHISK algorithm	34
Figure 4.3	Base Rules for anchoring each slot	35
Figure 4.4	Example WHISK Rule	36
Figure 4.5	Two example strings for Unique Match Sequence	36
Figure 4.6	Unique Match Sequence of two sample strings	37
Figure 4.7	Rule Representation created from Unique Match Sequence	37
Figure 5.1	Displaying results for event-based querying	40
Figure 5.2	Player-based querying	41
Figure 5.3	The System Architecture	42
Figure 5.4	System Overview	43
Figure 5.5	XQuery Example for Corner Event	44
Figure 5.6	XQuery Example for Goal Event	44
Figure 5.7	XQuery Example for Foul Event	45
Figure 5.8	XQuery Example for Substitution Event	46
Figure 5.9	Stages of JMF	48

LIST OF TABLES

TABLES

Table 3.1 Match Events	29
----------------------------------	----

CHAPTER 1

INTRODUCTION

There is a rapid growth of multimedia content available on broadcast and internet. Nevertheless, the need for accessing video anywhere and anytime is increased. The current technological developments present appropriate ways of saving and querying video files that include movies, news, sports events, medical scenes and security camera recordings. There are mainly two problems in the implementation of video archives: The first problem is related to the modeling and storage of videos and their metadata information. The second problem is how to find the metadata related to the video.

Multimedia databases stretch out usual databases for their data management. Metadata should be managed with media data for content based retrieval. However there are some difficulties that have to be concerned for multimedia retrieval. For instance data volume of media objects is very high, media objects need to be annotated, there are different kinds of file formats that have to be supported and multimedia objects could be complex such that they can be combination of several objects. Many different multimedia database systems have been developed and all of those database systems use different technologies for extracting the information from multimedia objects.

Content based knowledge extraction from large multimedia repositories is an important research area. There are three broad categories of annotation techniques that are used for multimedia repositories: manual annotation, image and video analysis techniques and extraction of metadata from text. Creating such metadata by hand is cumbersome and not practical for digital videos. On the other hand, there are image processing techniques that make it possible to analyze scenes or audio pausing. These are still technically challenging areas. Instead of video content, event detection can be done by using the textual information accompanying the video. However it

is not always possible to find textual data with all of the video types. For some domains such as football domain, movie domain and news domain, there are textual and semi-structured data sources that can be used for information extraction for videos.

There exists some recent work on metadata extraction for sports events [3, 14, 21, 22]. These projects target the soccer domain and event extraction from match reports and some of them [21, 22] make the alignment of extracted information with video content. However all of these projects store the match events in their specialized format and make the alignment according to their specific data representation.

1.1 Motivation

This thesis focuses on metadata extraction from natural texts in soccer domain and MPEG-7 mapping of this extracted information. We present a system that annotates soccer game videos automatically by using the information in match summary texts. The proposed system downloads live match reports from UEFA by a web-crawler. It, then, tags these match reports by regular expressions. All events are extracted from match reports via a hand-written rule set. In addition to the experiment with hand-written rules, two different information extraction algorithms, WHISK [18] and Event Extraction with Unique Match Sequence [4], are implemented on the same match corpus in order to analyze the results, compare them and make the event extraction more automatic. These extracted events are converted to valid MPEG-7 files and match corpus is generated. A user interface is provided for querying and searching the match corpus. Relevant video segments of the game are displayed for successful search results.

1.2 Contributions of the Thesis

The contributions of our approach include the mapping of match events into the MPEG-7 standard [34]. The usage of MPEG-7 standard makes our corpus interoperable with other systems. Besides that, synchronizing match events and match videos with MPEG-7 standard provides convenience. The search and query operations on MPEG-7 files are managed by XQuery language. Since there are many search options, we obtain a dynamic XQuery generator over MPEG-7 files. If a match video is found as the result of querying, it could be displayed according to the time of the event. The

synchronization of video and event is accomplished by the minute information of event that is gathered from the MPEG7 file. Another contribution of the thesis is the use of Event Extraction with Unique Match Sequence [4] algorithm in information extraction for the first time.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 gives insight information about current metadata extraction from soccer domain and alignment of video and metadata. Chapter 3 explains how the match corpus is created and how information is extracted from that corpus and that information is mapped to MPEG-7. In Chapter 4, supervised learning algorithms, WHISK and Event Extraction with Unique Match Sequence are presented. In Chapter 5, searching the match corpus and alignment of related information with video is analyzed. Finally in Chapter 6, conclusions and possible future extensions are listed.

CHAPTER 2

RELATED WORK

2.1 Information Extraction

There is an explosive growth in the amount of information with the advances in computer technology and popularization of the Internet. The vast amount of that information is accessible to an ordinary person. However, it is not easy for the user to find the related piece of information that he needs. This results with a new problem such as finding and extracting the necessary information. Information Extraction (IE) can be used for analyzing the unstructured texts and extracting the related information into a structured representation. It does not attempt to understand whole document, only transforms free text into structured form. In this section, we cover the IE approaches and methods of previous research, first in general then in soccer domain.

2.1.1 Information extraction from text

According to their learning methodology, automatic IE approaches are divided into three classes: supervised, semi-supervised, and unsupervised. In the supervised approaches, the learning process uses the annotated text segments and the syntactic structure of the surrounding text. There are many different supervised IE systems, and we review some of them here.

The WHISK system [18] is a learning system that extracts rules for three different types of documents: structured, semi-structured and free text. Its extraction patterns are similar to the regular expressions. It can generate both single-slot and multi-slot patterns. The details of the algorithm are given in Chapter 4.

The CRYSTAL system [19] is a learning system that generates a concept dictionary from annotated training data. It starts from initial nodes and tries to merge the

similar nodes by relaxing the constraints. The merged nodes are verified against the all documents. This process continues until no other merge can be executed.

The SRV system [8] is a learning system that is based on relational learning procedure. It starts with entire set of examples since it is a top-down process. Example space is divided into two example sets, negative examples and positive examples. SRV tries to extract positive examples. At each step, it adds predicates to cover more positive examples.

Boosted Wrapper Induction (BWI) [9] is a trainable IE system that performs information extraction in both free text and structured text. BWI learns extraction rules composed of simple patterns. It classifies each token a boundary that marks the beginning or end of a field and builds up a pattern from these training sets.

2.1.2 Information extraction in soccer domain

Information extraction from different kinds of sources has been popular recently. Especially for multimedia content annotation, information extraction can be preferred over video analysis if an associated text is available with the multimedia data. For instance, it is easy to find text describing videos in sports domain, specifically soccer videos. Popular sport sites publish match reports in a structured or semi structured format where events of the game are summarized along with their time information. The video segments can be annotated by aligning them with the time information and extracting the metadata from the text in summaries.

There exist many projects that have been developed for metadata extraction for sports events. Xu et. al. [22] provide a framework for detecting events from live sport videos and also live text analysis. They have four modules that are live text/video capturing, live text analysis, live video analysis and live text/video alignment. Live text analysis module extract the events from text and then these event will be synchronized with video with the live text/video alignment module. It regularly gets the HTML or Flash files and checks the difference if there is an update on text. Game events are extracted from these files by rule-based matching. Also for all of the sports activities there are tables that contain keywords. For instance for soccer domain they have keywords such as goal, shot, save and offside. Since the structure of text is well defined, events can be extracted by keyword matching. There are two approaches for the alignment of video and extracted event: game start detection and digital clock in the video. After finding the time of events and the corresponding time in that video,

alignment can be done easily.

Yankova and Boytcheva [23] accomplish information extraction by template pattern matching in football area. They use GATE which is a machine learning tool, for intermediate analysis results. GATE tool outputs logical forms that the system can recognize and then the system uses direct matching of these forms with the templates that are prepared.

The system described in [7], using a priori semantic knowledge, is an automatic audio video summarization tool by using the content-based metadata. The minute by minute match reports are downloaded and events are extracted from these text files by parsing manually. Then these events are classified as the ones take part in television highlights and the ones that do not take part. Then results are compared such that goal event is displayed in highlights with 100 percentage but shots on goal is displayed in highlights with 55 percentage.

The system described in [21] presents a framework for semantic annotation, indexing and retrieval of sports video. In this framework, they combined web-casting text analysis, broadcast sports video analysis and text/video alignment to extract events and detect event boundaries. A keyword matching routine is used during event extraction. A generic visual feature representation for modeling the event structures is proposed. On the other hand, they have also another module which finds the exact time of video by digital clocks in the scenes. Therefore the alignment can be done easily by that module for user specific queries. User can make queries related to game, event, player and team.

SOBA [3] is an ontology based approach for metadata extraction from match reports in soccer domain. SOBA automatically downloads match reports from UEFA and FIFA and sends them to a linguistic annotation web service. An ID is given to each match that is available in the web sites. Named entity recognition and part of speech tagging techniques are used while extracting events from linguistically annotated match reports. After extracting events, they are mapped to ontology that is provided by SmartWeb [34].

The system described in [14] aims to extract events from either tabular match reports which are structured and minute by minute match report which is unstructured. They interoperate the event time point for the inconsistency of the information in text and the actual video.

Compared with previous approaches, the contributions of our approach include the

mapping of match events into the MPEG-7 ontology. The usage of MPEG-7 standard makes our system interoperable and convenient. In addition, two supervised learning algorithms are used on our match corpus and compare their results.

2.2 Data Models for Multimedia Databases

2.2.1 XML Databases

XML is a software and hardware independent tool for carrying information. It was developed to mark up content but it also became an important part of data storage. Using XML as data storage has some advantages that are platform independency, human-readable format which makes it easier to fix errors, extensibility that without breaking the older versions adding extra information and large number of third party tools for processing XML documents [16].

XML Structure

There should be specifications for the structure of XML documents, since XML is a method to describe structured data. Document Type Definitions (DTD) and XML Schemas are two different ways to specify valid elements in a document according to certain aspects of these elements. XML documents can be valid or invalid according to a DTD or schema. The purpose of XML schemas is to define the nature and the component parts of XML markup constructs. These express syntactic, structural and value constraints for document instances. Figure 2.1 shows an example XML fragment.

```
<player name="Eric Cantona">  
<team>Manchester United</team>  
<age>32</age>  
</player>
```

Figure 2.1: XML Fragment

Document Type Definitions(DTD) DTDs were the original means of specifying the structure of an XML document. They are different from XML documents and they are used for specifying the order and occurrences of elements in XML document.

Figure 2.2 shows an example DTD for the XML fragment Figure 2.1. In that DTD specification, player element has two child elements that are team and age. It has one attribute element that is name. Both of the elements and attributes contain character data. PCDATA means parsed character data that is text found between the start tag and the end tag of an XML element. CDATA also means character data but not a parsed character data.

```
<!ELEMENT player (team, age)>
<!ATTLIST player name CDATA>
<!ELEMENT team (#PCDATA)>
<!ELEMENT age (#PCDATA)>
```

Figure 2.2: XML DTD

XML Data Reduced(XDR) For a number of reasons, DTDs become insufficient for the needs of XML users. This is because of their different syntax and their unsupported data types. XDR overcame some inadequacies about the DTDs by supporting a number of data types and relational database management systems. However it eventually rejected.

```
<Schema name="playerschema" xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="team" dt:type="string" />
  <ElementType name="age" dt:type="ui1" />
  <AttributeType name="name" dt:type="string" />
  <ElementType name="player" order="seq">
    <element type="team" minOccurs="1" maxOccurs="1"/>
    <element type="age" minOccurs="1" maxOccurs="1"/>
    <attribute type="name" />
  </ElementType>
</Schema>
```

Figure 2.3: XML XDR

In Figure 2.3, team and age are the element type and name is the attribute type. The player element is composed of team and age elements and name attribute. Team element and name attribute has string content, but age element has unsigned integer as its content.

XML Schema Definitions(XSD) The W3C XML schema recommendation provides description schema that defines the structure and constraint model of XML. XSD supports more data types compared to XDR. It also supports the creation of custom data types and object oriented concepts like inheritance and polymorphism. The example given in Figure 2.4 is the same specification schema as described in Figure 2.2 and Figure 2.3.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" >
  <element name="player">
    <complexType>
      <sequence>
        <element name="team" type="string"/>
        <element name="age" type="unsignedInt"/>
      </sequence>
      <attribute name="name">
        <simpleType>
          <restriction base="string">
            <pattern value="[A-Za-z]*"/>
          </restriction>
        </simpleType>
      </attribute>
    </complexType>
  </element>
</schema>
```

Figure 2.4: XML XSD

In this XSD specification, player is a complex type, and it has child element types. The player element is composed of team and age elements together with name attribute. Team and name have string type of content and age has unsigned integer type content. Besides that, name should match the regular expression that any character followed by any number of characters.

Managing XML Documents

There are three common ways of managing XML files: keeping them in a file system, in a relational database or native databases. The first two ones limit the functionality of XML.

For small applications, keeping files in a file system can be preferable. The system commands can be used for searching, querying or modifying the content.

Another alternative way is keeping documents in relational databases. This is

a better way of data management since this model provides transactional control, security, multi-user access. Besides that, most of the relational databases supports full text search. Elements and attributes of XML document can be indexed. Document is searched if an index is applied. If an instance is found, it will be recorded to index table with the identifier of the document instance.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
<element name="player">
<complexType>
  <sequence>
    <element name="name" type="string"/>
    <element name="surname" type="string"/>
  </sequence>
  <attribute name="player_no">
    <simpleType>
      <restriction base="string">
        <pattern value="\d*\d"/>
      </restriction>
    </simpleType>
  </attribute>
</complexType>
</element>
</schema>
```

If we want to index the element player_no we can store the indexes in the following tables:

Player Numbers:

PlayerNo	INTEGER
DocumentId	INTEGER

Documents:

DocumentId	INTEGER
Content	LONGVARCHAR

Figure 2.5: XML Document as a Column of a Table

In Figure 2.5 when a document is inserted to the database the application adds the document to the Documents table. It searches the content of the document for the attribute playerno and records the found values of the attribute and the documentId of the document to the Student Numbers table.

A native XML Database provides a model for XML document and uses that model for storage and query operations. This type of databases preserves the order of documents and comments. Therefore they can be used with document-centric systems and

support the XML query languages [24].

Querying with XML

Extracting data from XML documents is an essential part of XML specification. There exist a number of languages for querying XML documents including Lorel, Quilt, UnQL, XDuce, XML-QL, XPath, XQL, XQuery and YaTL. XPath [6] is already a W3C recommendation but XQuery [13] is on its way.

XPath XPath is a language for addressing parts of an XML document and is used for selecting portions of an XML document. Its syntax is similar to regular expressions and is designed to operate on a single XML document. There are some examples in Figure 2.6 for a better understanding of XPath.

- **/player/team**
Selects all team elements that are children of root element player
- **//age**
Selects all age elements in the document.
- **/player/***
Selects all child elements of the root element player
- **/player/@name**
Selects all name attributes of the player elements in the document
- **//*[name()='team']**
Selects all elements that are named "name"
- **count (//team)**
Find the count of all team elements that appear anywhere in the document
- **//player[team = "Manchester United"]**
Selects all player elements, which have a team child element of value "Manchester United"
- **//player[@name = "Eric Cantona"]**
Selects all player elements, which have a name attribute of value "Eric Cantona"
- **(team | age)**
Selects current context node that has team or age child nodes

Figure 2.6: XPath Example

XQuery Xquery is designed to afford the same functionality and underlying formalism as SQL does for relational databases. World Wide Web Consortium developed XQuery to query collections of XML and anything that can appear as XML. Unlike XPath, XQuery supports the manipulation of documents. It is also not only designed to operate on a single document [13].

The features of XQuery are explained with examples for each type in the following sections.

Path expressions: XQuery supports path expressions that are used in XPath.

```
document("Arsenal-Liverpool.xml")//player/@name = "Henry"
Find the player element and name attribute of that
element is "Henry"

document("Arsenal-Liverpool.xml")//player/team = "Arsenal"
Find the player element that has a team child element
and is "Arsenal"
```

Figure 2.7: XQuery Path Expression

Element Constructors: XQuery enables the creation and generation of the elements by queries. These elements can be put into a query in an expression called an element constructor.

```
<player name = {$name}>
    {$team}
    {$age}
</player>
```

Figure 2.8: XQuery Element Constructor

The expression in Figure 2.8 generates a player element with name attribute. Name, team and age are the attributes that will be assigned to player element.

FLWOR Expressions: FLWOR stands for For, Let, Where, Order by, and Return keywords. It is a query expression that is composed of these keywords.

1. FOR clause is an iteration construct over the set of results returned by a query.

2. LET clause is assigning variables to values.
3. WHERE clause contains logical expression that filters the results of LET and FOR clauses.
4. ORDER BY clause is for sorting the results according to the given clause.
5. RETURN clause is for the output of the query that produces the result set.

```
FOR $player IN document("Arsenal-Liverpool.xml")//player
WHERE $player/team = "Arsenal" AND $player/age = 22
RETURN $player/@name|
```

Figure 2.9: XQuery FLWOR Expressions

The expression in Figure 2.9 returns a list of player names whose age is equal to 22 and team is Arsenal.

Conditional Expressions: A conditional expression works as a test expression that has two results true and false. If the value of test expression is true first expression will be evaluated otherwise second one will be evaluated.

```
FOR $player IN document("Arsenal-Liverpool.xml")//player
RETURN
<result>
{
    $player/@name,
    IF ($player/age < 20)
    THEN $player/position
    ELSE $player/age
}
</result>
```

Figure 2.10: XQuery Conditional Expressions

The expression in Figure 2.10 returns a list of players. If the age of player is smaller than 20, it includes the position information of the player, if age of player is greater than 20, it includes the age information of the player.

Quantified Expressions: XQuery has constructs similar to quantifiers used in mathematics and logic. SOME clause is for testing if there is at least one node satisfying the related predicate. EVERY clause is used for testing if all of the values adjust the predicate.

```
FOR $player IN document("Arsenal-Liverpool.xml")//player
WHERE SOME $player IN $playervariable
    (contains playervariable/team = "Arsenal")
RETURN $palyer/@name
```

Figure 2.11: XQuery Quantified Expressions

The expression in Figure 2.11 returns the list of player names, whose team has the word "Arsenal".

User Defined Functions: XQuery supports the user defined functions apart from XPath. The expression in Figure 2.12 finds the maximum depth of a given element.

```
NAMESPACE xsd = "http://www.w3.org/2001/XMLSchema"
DEFINE FUNCTION findMaxDepth($player) RETURNS xsd:integer
{
    #Recursively find the depth of element
    IF (empty($player/*)) THEN 1
    ELSE max(depth($palyer/*)) + 1
}
```

Figure 2.12: XQuery User Defined Functions

2.2.2 MPEG-7

In recent years, the usage of multimedia content is increased. Therefore standardization of the multimedia content is an essential part for scalability, intelligence and interoperability. MPEG-7 has been developed for that purpose. It is based on the common used data description layer XML. Using XML technology, MPEG-7 can describe low-level visual and audio features as well as high-level features such as semantics.

MPEG-7 standard deals with a wide range of multimedia properties such as:

1. Visual features
2. Audio features
3. Structure
4. Semantics
5. Management
6. Collection organization
7. Summaries
8. User Preferences

MPEG-7 separates the multimedia content data from the description and presentation. For images, spatial decomposition, for audio files temporal decomposition and for videos temporal, spatial or spatio temporal decomposition are used. MPEG-7 provides a description scheme for the global information like title, creator and genre whereas it provides descriptors and data types for the description of color, texture, shape and motion.

Overview of MPEG Standards

MPEG (Motion Picture Expert Group) is a working group that aims developing standard for compression, decompression and processing video and audio. It has proposed several MPEG standards [30].

The MPEG-1 has been designed for digital storage of videos and its quality is similar to TV. It targets the video transmission over low bandwidth networks such as telephone cables. VideoCD standard is based on MPEG-1 and enables the storage of 74 minutes audio and visual data.

MPEG-2 was developed for digital televisions. It aims to become a standard for digital televisions with high data rates. Furthermore, it extends the MPEG-1 with resolutions up to 16383x16383 pixel and supports up to 5 audio channels.

MPEG-4 was developed for multimedia applications. It does not define a compression method but it stores the advanced tools for compression algorithms for audio and

visual data. As an internal language JavaScript is used and also MPEG-4 provides a java interface layer. Therefore java classes could be used within MPEG-4 content.

MPEG-4 describes the integration of objects into video stream [28]. It is designed for the usage in internet, wireless video, simulation and training. The data can be organized in a tree view. For that reason, each object can be separately compressed. Scenes are generated by these corresponding objects. The final description of MPEG-4 is the interaction between the user and scene. For instance user can select the language of a video. MPEG-4 is a content based description of a multimedia file.

The next standard by MPEG group is MPEG-7 [29]. It enables the storage of metadata descriptions of multimedia files. MPEG-7 is designed for multimedia content and could be used in a wide area of multimedia content, such as:

- Organization of audiovisual databases
- Broadcast media
- Multimedia Services
- Digital Libraries
- Personalized Services
- Digital photo/video albums

MPEG-7 can describe pictures, graphics, audio and visual contents; also it is possible to describe a content that is not audio and visual. It can be independently used from video. Using MPEG-7 it is possible to represent data that is defined with MPEG-4. MPEG-7 descriptions can be used to increase the performance of other MPEG standards. We use MPEG-7 formalism to store videos in this thesis.

MPEG-21 targets to develop a framework for multimedia applications [27]. It is XML based and aims to communicate machine-readable license information.

Introduction to MPEG-7

MPEG-7 is used to describe metadata for user interaction. The description generation or retrieval is not the issue of MPEG-7. These components are application dependent that each developer uses its own [10]. The technical description part gives a brief information about the MPEG-7 standard and the MPEG-7 descriptors section explains all descriptors used in technical realization.

Technical Description

MPEG-7 is based on XML schema. The main elements are Descriptors (D), description Schemes (DS), a description definition language (DDL). Descriptors specify the single features such as syntax and semantics of objects like colors, speech or persons. Description Schemes is used for defining the relationship between components such as Descriptors or other description schemes.

The Description Definition Language allows the creation or extension of MPEG-7. A DDL schema contains the constraints for a valid MPEG-7 description [29]. It also enables users to create simple and complex types. Simple types are data type constraints but complex types are structural constraints. Furthermore, DLL enables using simple and complex types like object oriented sense.

Descriptors can be used for low level features such as title and creator and high level descriptors such as objects and the relationship between these objects. Each application that uses MPEG-7 uses a subset of descriptors that are defined in MPEG-7 standard. If these are not sufficient for their needs, they have to define their own descriptors. In Figure 2.13 the basic elements that MPEG-7 defines can be seen.

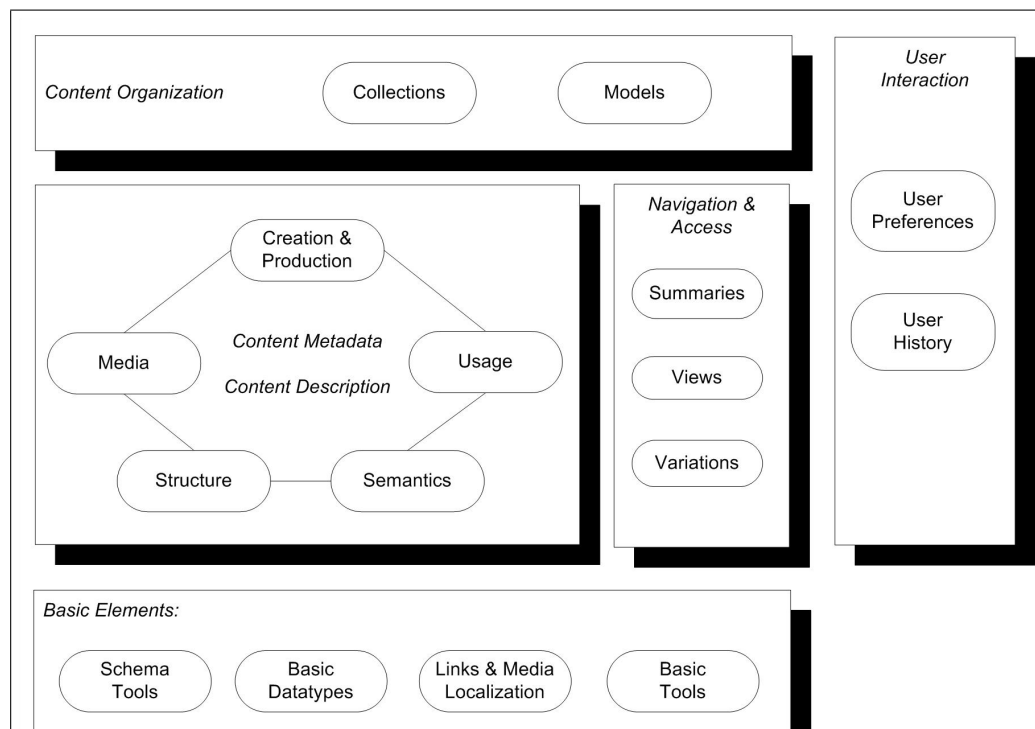


Figure 2.13: MPEG-7 area of application. The figure is based on the MPEG-7 standard [5]

There exist MPEG-7 System tools for generating and evaluating audiovisual descriptors. These tools provides synchronizing and transmission of coded representations of MPEG-7 data. A lot of descriptions are submitted to MPEG-7 and some of them are accepted and included. MPEG-7 standard can be extended by the application requirements or each application selects which part of MPEG-7 standard to use.

MPEG-7 Descriptors

MPEG-7 standard has its own descriptors for describing low level features. These descriptors are defined by MPEG group experts. In Figure 2.14 overview of the MPEG-7 visual features can be seen. The visual descriptors specify feature parameters that can be used for similarity matching of a compressed representation. Most of descriptors can be used directly for images. However some kinds of descriptors need preprocessing for the input. MPEG-7 specifies semantic descriptor for a high level of abstraction.

This section describes the description that we use in this thesis. The complete description of all visual descriptors can be found on the visual description part of the of the MPEG-7 standard.

Semantic Descriptor Semantic meaning of a multimedia content is described under semantic descriptor. It enables users to define objects or use the predefined objects, and define relationship between these objects in terms of mapping real life situations. MPEG-7 has some entities that are referenced from multimedia content such as objects, agent objects, events, concepts, states, places and times.

In Figure 2.15, SemanticBase element that identifies the semantic entities is illustrated. All of the content is not shown here for simplicity/readability purposes. The object describing, Morientes is created with an agent type. Also that object has a relation with the goal event.

In Figure 2.16, The graph element groups a set of relations. A relation consists of a name (type attribute), a source entity (source attribute) and target entity (target attribute).

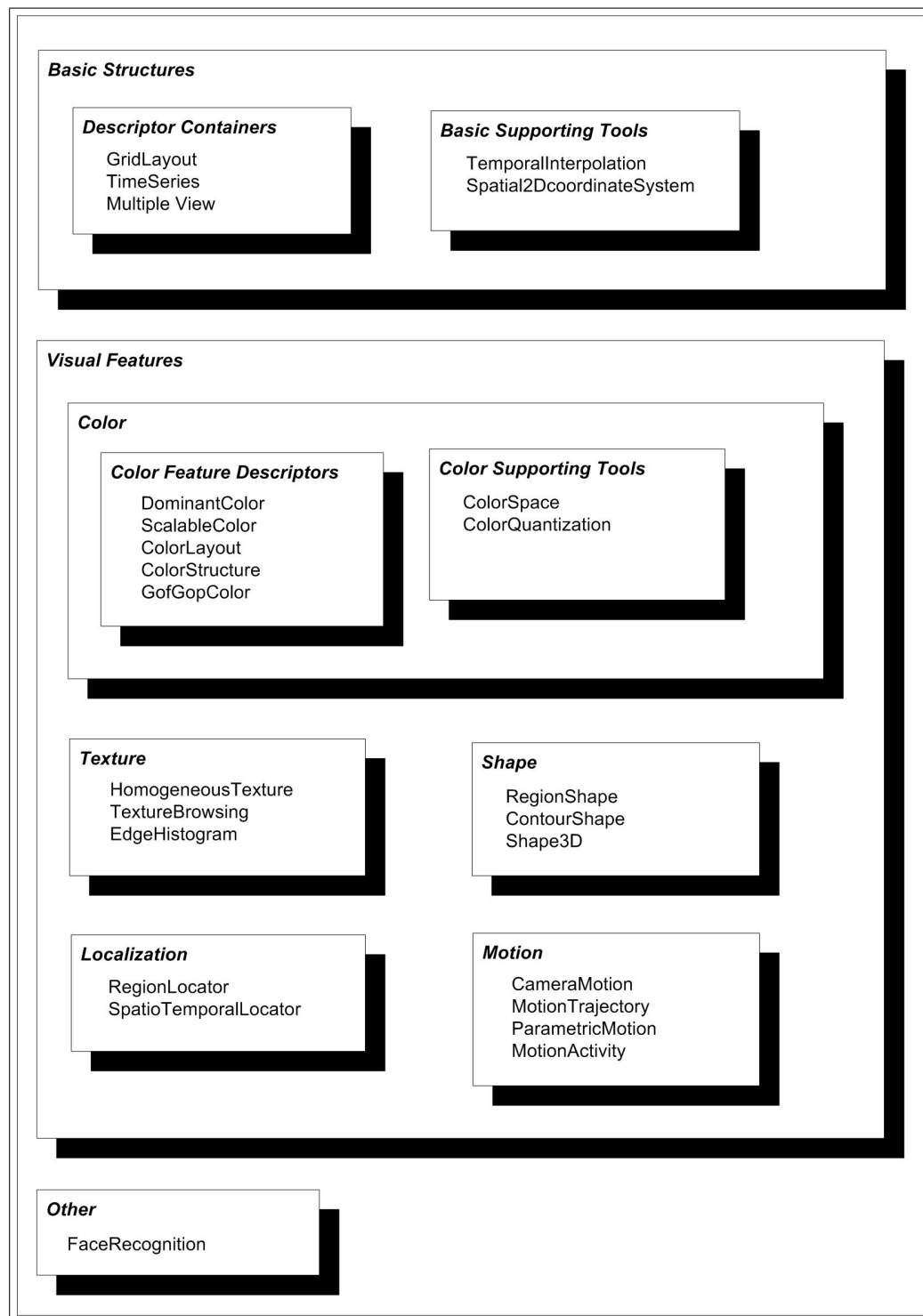


Figure 2.14: Overview of the MPEG-7 visual features. The figure is based on the MPEG-7 standard [5]

```

<SemanticBase id="Morientes-object" xsi:type="AgentObjectType">
  <Label href="#">
    <Name>Soccer player</Name>
  </Label>
  <Definition>
    <FreeTextAnnotation>
      Spanish soccer player named Javier Morientes</FreeTextAnnotation>
    </Definition>
    <Relation type="urn:...:agentof" target="#Goal-event"/>
    <Agent xsi:type="PersonType">
      <Name>
        <GivenName>Javier</GivenName>
        <FamilyName>Morientes</FamilyName>
      </Name>
    </Agent>
  </SemanticBase>

```

Figure 2.15: Semantic Description Example

```

<Semantics>
  <Graph>
    <!-- Morientes kicks the ball towards the goal -->
    <Relation type="urn:...:agentof" source="#Morientes-ob" target="#kick-ev"/>
    <Relation type="urn:...:patientof" source="#Ball-ob" target="#kick-ev"/>
    <Relation type="urn:...:destinationof" source="#GoalArea-ob" target="#kick-ev"/>
    <!-- Goalkeeper makes an error -->
    <Relation type="urn:...:agentof" source="#Svensoon-ob" target="#Error-ev"/>
    <!-- Morientes is a player of the Spanish soccer team -->
    <Relation type="urn:...:memberof" source="#Morientes-ob" target="#SpanishTeam-ob"/>
    <!-- The goalkeeper is a player of the Swedish team -->
    <Relation type="urn:...:memberof" source="#Svensoon-ob" target="#SwedishTeam-ob"/>
    <!-- Location and time of the goal event -->
    <Relation type="urn:...:hasLocationof" source="#Goal-ev" target="#Santberna-ob"/>
    <Relation type="urn:...:hasTimeof" source="#Goal-ev" target="#goal1-time"/>
  </Graph>
</Semantics>

```

Figure 2.16: Semantic Description Example

CHAPTER 3

METADATA EXTRACTION FROM TEXT

There exists an enormous amount of information in natural language texts. This information has to be mapped into structural form, to analyze and process it. Metadata Extraction systems extract parts of information by applying predefined structured representation to natural language texts. In our system, natural language texts are minute by minute match reports that reside in UEFA web site and match events will be the extracted information. In order to generate match corpus, minute by minute match reports are downloaded from UEFA web site. Then these match reports are tagged by using regular expressions and match events, that are decided in hand written rule sets, are extracted and stored in a XML database. Finally, these XML databases are mapped into MPEG-7 files for each match report.

Figure 3.1 shows the overview of the system which is implemented for metadata extraction. The system architecture consists of the Crawler, Entity Extractor, Match Event Extractor and MPEG-7 generator modules. The crawler module is responsible for parsing and crawling the UEFA match centre web sites and finding the Champions League and UEFA cup match links, and downloading the necessary minute by minute reports into the specified local drive. The Entity Extractor module annotates these match reports using regular expressions and then the Event Extractor module extracts events from tagged match reports by using hand written rules as an XML file. The MPEG-7 generator module generates MPEG-7 files from the extracted events.

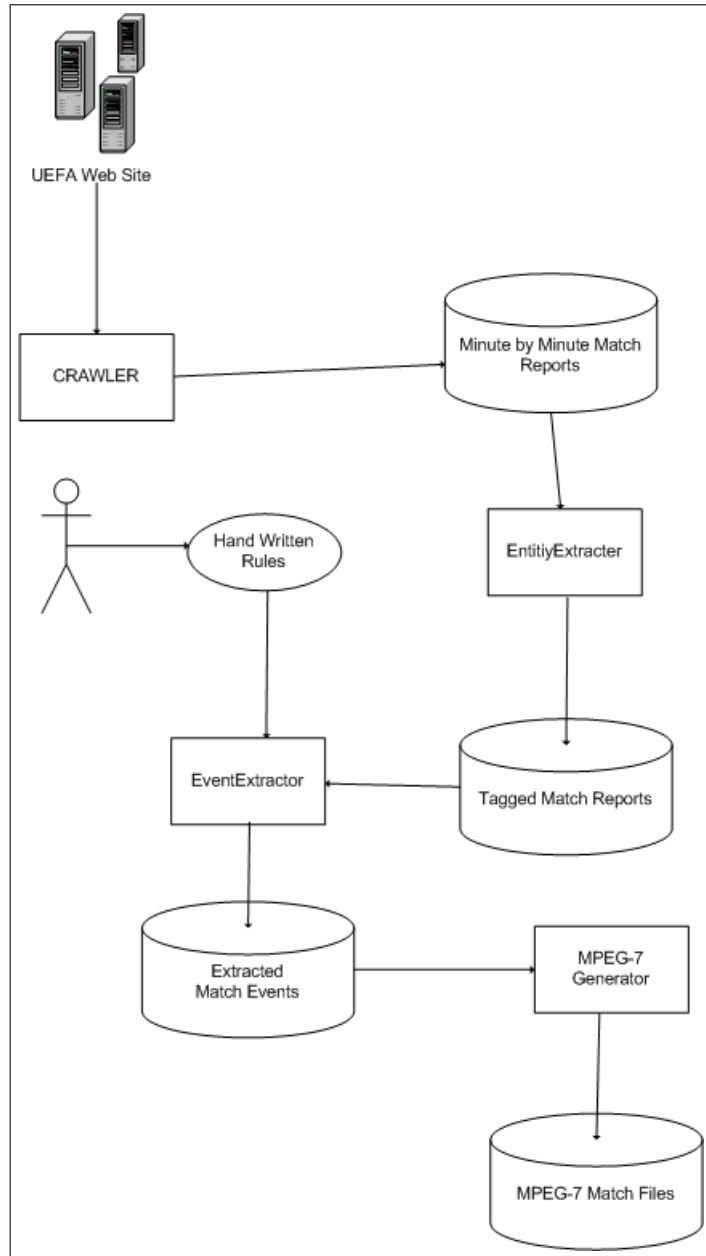


Figure 3.1: System Overview

3.1 Web-Crawler

The web-crawler is used to search the Internet (UEFA site) and produce the match corpus. The match data is formed from HTML documents of official web site of UEFA. There are two big organizations that UEFA arranges, UEFA Champions League and UEFA Cup. These organizations have their own web sites and match centers. In their match centers, the data source contains for each match, semi-structured data of player names, referees, match result and scorers of the match. Minute by minute

match report is also provided at the match center. Minute by minute match reports inform people about the events at the game in a textual form. These reports include the events, performers of events and their exact time point.

The crawler is able to extract minute by minute match reports from Champions League and UEFA Cup match centers. At the fixtures and results part of each competition, there are links to the match days. Figure 3.2 shows the fixtures and results section of UEFA Champions League. At the right side of the page, there are round names in bold. They all provide links to the rounds that have match reports.

UEFA CHAMPIONS LEAGUE

News | **Fixtures & Results** | Standings | Clubs | Statistics | Finals & Format | History

T-T
The UEFA Champions League comprises of three qualifying rounds, a group stage, and four knockout rounds.

Qualifying
In matches in the three qualifying rounds, clubs play two matches against each other on a home and away basis, with the club scoring the greater aggregate of goals qualifying for the next round. In the event of both teams scoring the same number of goals, the team which scores more goals away qualifies.

Group stage
The 16 winners of the third qualifying round ties join a similar number of automatic entrants in the 32-team group stage. The clubs are split into eight groups of four teams, who play home and away against each of their pool opponents between September and December to decide which two teams from each pool advance to the first knockout round. The third-place finishers in each pool enter the UEFA Cup Round of 32.

Knockout phase
From the last 16 until the semi-finals, clubs play two matches against each other on a home and away basis with the same rules as the qualifying rounds applied. In the last 16, group winners play runners-up other than teams from their own pool or nation, while from the quarter-finals on the draw is free.

Final
The final is decided by a single match - which this year will be played in Rome.

Rounds

- First qualifying round**
15 - 23 July 2008
- Second qualifying round**
29 Jul - 06 Aug 2008
- Third qualifying round**
12 - 27 August 2008
- Group stage**
16 Sep - 10 Dec 2008
- First knockout round**
24 Feb - 11 Mar 2009
- Quarter-finals**
07 - 15 April 2009
- Semi-finals**
28 Apr - 06 May 2009
- Final**
27 May 2009

[Full season matches »](#)

MasterCard

football fans

Figure 3.2: Fixtures and Results of UEFA Champions League

In each of match days, there are links to the match reports. Therefore, match report links could be found by crawling from fixtures and results site. For each match report link, minute by minute reports are extracted and saved to the match corpus. Figure 3.3 shows the first qualifying round matches. For each match there is a link at the right side. Figure 3.4 shows an example minute by minute match report.

UEFA CHAMPIONS LEAGUE				
News Fixtures & Results Standings Clubs Statistics Finals & Format History				
First qualifying round				
Date	Home	Score	Away	
23/07/08	Pyunik	15:30	Anorthosis	More »
23/07/08	Levadia	17:30	Drogheda	More »
23/07/08	Budućnost Podgorica	20:00	Tampere United	More »
23/07/08	Modriča	20:00	Dinamo Tirana	More »
23/07/08	Göteborg	20:00	Murata	More »
23/07/08	Artmedia	20:15	Valletta	More »
23/07/08	Domžale	20:45	Dudelange	More »
23/07/08	Dinamo Zagreb	20:45	Linfield	More »
23/07/08	Valur Reykjavík	21:15	BATE	More »
22/07/08	Rabotnicki	17:00	Inter Bakı	More »
22/07/08	Ventspils	17:00	Llanelli	More »
22/07/08	Kaunas	18:30	Santa Coloma	More »
22/07/08	Sheriff	19:00	Aktobe	More »
22/07/08	NSÍ	20:00	Dinamo Tbilisi	More »
16/07/08	Dinamo Tbilisi	3-0	NSÍ	More »
16/07/08	Aktobe	1-0	Sheriff	More »
16/07/08	Drogheda	2-1	Levadia	More »
16/07/08	Linfield	0-2	Dinamo Zagreb	More »
15/07/08	Inter Bakı	0-0	Rabotnicki	More »
15/07/08	Tampere United	2-1	Budućnost Podgorica	More »
15/07/08	BATE	2-0	Valur Reykjavík	More »
15/07/08	Anorthosis	1-0	Pyunik	More »
15/07/08	Dudelange	0-1	Domžale	More »
15/07/08	Llanelli	1-0	Ventspils	More »
15/07/08	Dinamo Tirana	0-2	Modriča	More »
15/07/08	Valletta	0-2	Artmedia	More »
15/07/08	Santa Coloma	1-4	Kaunas	More »
15/07/08	Murata	0-5	Göteborg	More »
Rounds				
■ First qualifying round 15 - 23 July 2008				
■ Second qualifying round 29 Jul - 06 Aug 2008				
■ Third qualifying round 12 - 27 August 2008				
■ Group stage 16 Sep - 10 Dec 2008				
■ First knockout round 24 Feb - 11 Mar 2009				
■ Quarter-finals 07 - 15 April 2009				
■ Semi-finals 28 Apr - 06 May 2009				
■ Final 27 May 2009				
Full season matches »				
Proud sponsor of the UEFA Champions League				
football fans				
FANZONE				

Figure 3.3: Qualifying Round of UEFA Champions League

3.2 Text Annotation

In minute by minute match report, an event is described by one sentence which contains the minute of the event, performers of the event and teams of the performers. The structure of these sentences is well-defined. Therefore, extracting events from sentences could be done by matching the sentences with a template that consists of labeled match events. Before extracting event types, the match report must be labeled (tagged). Regular expressions are used to identify minutes, player names and team names.

In UEFA match reports minute information takes part at the beginning of the sentence and team information follows a player name. Player names are proper nouns and always start with uppercase letters. The team of a player is indicated by a team name in parentheses. The minute information is represented by simple numeric values. However for extra time of the match, numeric value is followed by a plus character and then another numeric value. For extra time after the match, prefix Ex. is used and it is followed by numeric values. The regular expressions for minute, extra minute, player name, score and team name are illustrated in Figure 3.5. All other words and








90		Maminov (in) - Gurenko (out) (Lokomotiv Moskva)
89		Agüero (Atlético) commits a foul after challenging Sennikov (Lokomotiv Moskva).
88		Dos Santos (Atlético) fouls.
88		Pelizzoli (Lokomotiv Moskva) makes a save.
88		Pernía (Atlético) delivers the free-kick on target.
87		Gurenko (Lokomotiv Moskva) gives away a free-kick following a challenge on Maxi Rodríguez (Atlético).
85		(3 - 3) Agüero (Atlético) scores! This dramatic game serves up a dramatic finale as Maniche threads a pass through to Agüero, who allows the ball to go past him before spinning his marker and then cleverly dinking the ball over the sprawling Ivan Pelizzoli to draw Atlético level and maybe earn them a share of the points.
83		Odemwingie (Lokomotiv Moskva) is adjudged to be in an offside position.
81		Maxi Rodríguez (Atlético) delivers the free-kick wide.
81		Cocis (in) - Samedov (out) (Lokomotiv Moskva)
80		Spahić (Lokomotiv Moskva) commits a foul after challenging Maxi Rodríguez (Atlético).
80		Pelizzoli (Lokomotiv Moskva) makes a save.
78		Luis García (Atlético) takes the corner.
78		Antonio López (Atlético) delivers the corner.
78		Pelizzoli (Lokomotiv Moskva) makes a save.
77		Maniche (Atlético) has an effort on goal.
77		Abbiati (Atlético) makes a save.
76		Sychev (Lokomotiv Moskva) has an effort on goal.
75		Maniche (in) - Jurado (out) (Atlético)
74		Agüero (Atlético) fouls.
73		Rodolfo (Lokomotiv Moskva) misses the target.
73		Samedov (Lokomotiv Moskva) takes the corner.
71		Pablo Ibáñez (Atlético) gives away a free-kick following a challenge on Odemwingie (Lokomotiv Moskva).
70		Jurado (Atlético) delivers the corner.
69		Luis García (in) - Simão (out) (Atlético)
68		Spahić (Lokomotiv Moskva) commits a foul after challenging Agüero (Atlético).
67		Ivanović (Lokomotiv Moskva) misses the target.
66		Antonio López (Atlético) is shown a yellow card.
66		Antonio López (Atlético) fouls.
65		Maxi Rodríguez (in) - Cléber Santana (out) (Atlético)

Figure 3.4: Example Minute by Minute Match Report

punctuation are labeled as token.

After a document containing match summaries is downloaded by the crawler, it is processed by tagging each sentence properly. The minute by minute text is transformed into an XML document where each sentence is represented as a separate element. The XML file contains labeled words and tokens under the sentence element. Converting plain match texts into structured XML files makes it easy to apply information extraction algorithms on the match corpus. Figure 3.6 shows a tagged form of the sentence: "87: Crouch (Liverpool) has an effort on goal."

On the other hand, while applying regular expressions, some exceptions are handled by the code. Some players are represented by their name and surname and some team names include two tokens. This problem is solved by applying the regular expressions

Minute:	[0-9]*[0-9]
Extra Minute:	[0-9]*[0-9][+]*[0-9]*
Player Name:	[A-Z]\D*
Team:	[()][A-Z]\D*[]]*.*
Score:	[()][0-9>[]]

Figure 3.5: Regular Expressions

```

<Sentence>
  <Minute>87</Minute>
  <PlayerName>Crouch</PlayerName>
  <Token>(</Token>
  <Team>Liverpool</Team>
  <Token>)</Token>
  <Token>has</Token>
  <Token>an</Token>
  <Token>effort</Token>
  <Token>on</Token>
  <Token>goal</Token>
  <Token>.</Token>
</Sentence>

```

Figure 3.6: A Tagged Sentence

and then checking the structure. For instance, if a player name is identified, before tagging that information as player name, the program examines the succeeding token. It is also a player name both of them together are tagged as a player name. In Figure 3.7, an example is shown for a tagged player name that has two tokens.

```

<Sentence>
  <Minute>65</Minute>
  <PlayerName>Martin Pedersen</PlayerName>
  <Token>(</Token>
  <Team>AaB</Team>
  <Token>)</Token>
  <Token>delivers</Token>
  <Token>the</Token>
  <Token>corner</Token>
  <Token>.</Token>
</Sentence>

```

Figure 3.7: Tagged Sentence with a two tokened player name

3.3 Match Event Extraction

In order to extract event information from the pre-processed texts we prepared a rule set for each soccer event. We have identified all distinct events appearing in match reports and identified different sentence structures for each of these events. For each event type there is a set of hand-written rules. These rules are applied to the data set to extract the events into another XML file for each match. The rule set can be thought as a template for match corpus events and the extracted information on that event. For each sentence in a match report, it compares the patterns of hand-written rules and if there is a match, it will extract the event from the rule set and fill it with the specified information in the sentence.

Figure 3.8 shows an example rule for discerning the corner event. Under the element *Rule* there are two sub elements *Pattern* and *MatchEvent*. *Pattern* is the template for the sentence describing the event. If the tagged sentence is coherent with the pattern of the rule, the corresponding event will be extracted according to the *MatchEvent* element of the rule. Template matching is done by matching the field name of the sentence with the *Pattern* sub-element of the rule element first and if the field name is *Token*, matching the content too. In Figure 3.8, there are three pieces of extracted data: minute, player name and team for corner event. This information is filled from the tagged sentence by matching the field names. After all sentences are scanned, an XML file is generated according to the extracted information from the rule set. The XML file contains the events in the format of *MatchEvent* element that is described under the *Rule* element. An example event representation can be seen in Figure 3.9. The rule set used to extract all events in a soccer game is given in APPENDIX A.

Table 3.1 lists all match events and the extracted fields for the events that we can process. For each of these events, all possible sentences are examined and rules that are similar to the one in Figure 3.8 are created. There are two important points in creating the rules: The first element *Pattern* is the template that will be matched with the sentences in match summaries and the second element *MatchEvent* represents the extracted information. As it is seen in Figure 3.8, the extracted information for corner event such as Minute, PlayerName and Team has no content. If this rule is matched with a sentence, they will be filled by gathering values for these fields from the input sentence.

```

<Rule>
  <Pattern>
    <Minute></Minute>
    <PlayerName></PlayerName>
    <Token>{</Token>
    <Team></Team>
    <Token>}</Token>
    <Token>delivers</Token>
    <Token>the</Token>
    <Token>corner</Token>
    <Token>.</Token>
  </Pattern>
  <MatchEvent>
    <CornerEvent>
      <Minute></Minute>
      <PlayerName></PlayerName>
      <Team></Team>
    </CornerEvent>
  </MatchEvent>
</Rule>

```

Figure 3.8: A Sample Rule for Extraction

```

<CornerEvent>
  <Minute>53</Minute>
  <PlayerName>Alonso</PlayerName>
  <Team>Bordeaux</Team>
</CornerEvent>

```

Figure 3.9: A Sample Event Representation

3.4 MPEG-7 Generation from Match Events

It is preferable that we store match events in a standardized manner so that other systems can use the same match corpus for their systems. Besides, we want match events to be synchronized with the football videos. For this purpose, we use MPEG-7 standard to keep the semantic annotations of the games.

In our work, an MPEG-7 file is created for each match that is extracted from the minute by minute match reports. In each MPEG-7 file, all events and their time points are represented according to MPEG-7 Descriptions Schemas. MPEG-7 standard allows us to define semantic content under the *Semantic Description*. *SemanticBase* element under the *Semantic Description* is used for performers of events with type *AgentObjectType* which let us to describe performer under the *Agent* element. *MediaTimePoint* under the *MediaTime* is used for the minute information of the event. For the team information, event and opponent team information, we used the *Relation* element. For event name, relation type will be *agentOf*; for team information of the

Table 3.1: Match Events

Match Events	Extracted Information
Cautioned	(Minute, PlayerName,Team)
Corner	(Minute, PlayerName,Team)
Foul	(Minute, FoulCommittedPlayerName, FoulCommittedTeam, FoulSufferedPlayerName, FoulSufferedTeam)
Free-Kick	(Minute, PlayerName,Team)
Goal	(Minute, PlayerName,Team)
FreeKickGoal	(Minute, PlayerName,Team)
PenaltyGoal	(Minute, PlayerName,Team)
OwnGoal	(Minute, PlayerName,Team)
GoalPosition	(Minute, PlayerName,Team)
Offside	(Minute, PlayerName,Team)
PenaltyEvent	(Minute, PlayerName,Team)
PenaltyMiss	(Minute, PlayerName,Team)
Redcard	(Minute, PlayerName,Team)
YellowCard	(Minute, PlayerName,Team)
Substitution	(Minute, SubstitutionInPlayerName, SubstitutionOutPlayerName,Team)
SaveGoal	(Minute, PlayerName,Team)

performer relation type will be *memberOf* and for the opponent team relation type will be *hasAccompaniedOf*. Player name is stored under the *Agent* element with type *PersonType*. Besides that, for each *SemanticBase* element an id is needed. In order to make unique ids, an id is created as follows: player name is followed by an underscore and the team of the player, and then another underscore, minute and the counter of event is appended.

There are some events that have hierarchical representations, such as foul event which is a combination of *FoulCommitted* and *FoulSuffered* event. For these events, two separate events are created and the relationship between these events is established by another *SemanticBase* element. The *SemanticBase* element that provides

```

<SemanticBase id="Prica_AaB_45+1_1" xsi:type="AgentObjectType">
  <Label>
    <Name />
  </Label>
  <Definition>
    <FreeTextAnnotation />
  </Definition>
  <MediaOccurrence>
    <MediaLocator xsi:type="TemporalSegmentLocatorType">
      <MediaTime>
        <MediaTimePoint>45+1</MediaTimePoint>
      </MediaTime>
    </MediaLocator>
  </MediaOccurrence>
  <Relation type="urn:...:agentOf" target="Cautioned" />
  <Relation type="urn:...:memberOf" target="AaB" />
  <Relation type="urn:...:hasAccompaniedOf" target="ANDERLECHT" />
  <Agent xsi:type="PersonType">
    <Name>
      <GivenName>Prica</GivenName>
      <FamilyName />
    </Name>
  </Agent>
</SemanticBase>

```

Figure 3.10: Example MPEG-7 Descriptors

```

<SemanticBase id="FoulEvent_28" xsi:type="AgentObjectType">
  <Label>
    <Name />
  </Label>
  <Relation type="urn:...:agentOf" target="FoulEvent" />
  <ObjectRef idref="Frutos_Anderlecht_89_26" />
  <ObjectRef idref="Olesen_AaB_89_27" />
</SemanticBase>

```

Figure 3.11: Example Foul Event MPEG-7 Descriptors

the relationship between two elements, is composed of a *Relation* element with type *agentOf* and two *ObjectRef* elements that refers to the id of two other events.

As it is seen in Figure 3.11, foul event has two object references to the *FoulSuffered* and *FoulCommitted* events. The id of foul event is the event name followed by the unique event counter number.

Figure 3.12 illustrates the commit foul event and Figure 3.13 illustrates the suffered foul event. *SemanticBase* id's of these events refer to the object reference id's of Figure 3.11.

In this chapter, we present our system that is based on event extraction by hand-written rules in soccer domain. In the following chapter, event extraction is done by two supervised learning algorithms, WHISK and Unique Match Sequence.

```

<SemanticBase id="Frutos_Anderlecht_89_26" xsi:type="AgentObjectType">
  <Label>
    <Name />
  </Label>
  <Definition>
    <FreeTextAnnotation />
  </Definition>
  <MediaOccurrence>
    <MediaLocator xsi:type="TemporalSegmentLocatorType">
      <MediaTime>
        <MediaTimePoint>89</MediaTimePoint>
      </MediaTime>
    </MediaLocator>
  </MediaOccurrence>
  <Relation type="urn:...:agentOf" target="FoulEventCommitted" />
  <Relation type="urn:...:memberOf" target="ANDERLECHT" />
  <Relation type="urn:...:hasAccompaniedOf" target="AAB" />
  <Agent xsi:type="PersonType">
    <Name>
      <GivenName>Frutos</GivenName>
      <FamilyName />
    </Name>
  </Agent>
</SemanticBase>

```

Figure 3.12: Example Foul Committed Event MPEG-7 Descriptors

```

<SemanticBase id="Olesen_AaB_89_27" xsi:type="AgentObjectType">
  <Label>
    <Name />
  </Label>
  <Definition>
    <FreeTextAnnotation />
  </Definition>
  <MediaOccurrence>
    <MediaLocator xsi:type="TemporalSegmentLocatorType">
      <MediaTime>
        <MediaTimePoint>89</MediaTimePoint>
      </MediaTime>
    </MediaLocator>
  </MediaOccurrence>
  <Relation type="urn:...:agentOf" target="FoulSuffered" />
  <Relation type="urn:...:memberOf" target="AAB" />
  <Relation type="urn:...:hasAccompaniedOf" target="ANDERLECHT" />
  <Agent xsi:type="PersonType">
    <Name>
      <GivenName>Olesen</GivenName>
      <FamilyName />
    </Name>
  </Agent>
</SemanticBase>

```

Figure 3.13: Example Foul Suffered Event MPEG-7 Descriptors

CHAPTER 4

SUPERVISED LEARNING FOR INFORMATION EXTRACTION

The need for information extraction from textual data is grown, as more and more textual data become available on-line. For this purpose, different Information Extraction (IE) algorithms are designed. IE systems range from structured texts with tabular information to free text. Extraction rules that identify related information to be extracted are key elements of Information Extraction systems.

In this thesis, as it is described in Chapter 3, event extraction is first done by hand written rules. However, hand written rules are domain specific and can not be used for other domains. In this chapter, two approaches for information extraction, WHISK and Event Extraction with Unique Match Sequence are put into practice on our match data to analyze the results, compare them and make the event extraction more automatic.

4.1 Whisk Algorithm

WHISK is a supervised learning algorithm that requires hand tagged instances [18]. Tagging process will be described later. WHISK can operate on structured, semi-structured and free text. When WHISK is applied to structured or semi-structured text, it does not require syntactic analysis. For free text, WHISK works best if the text is tagged by a syntactic analyzer. For structured textual data, the rules are represented by fixed order of related information that can be HTML tags that delimit strings to be extracted. On the other hand, for free textual data, there are several steps that can be syntactic analysis, semantic tagging, and recognizers for domain specific objects. Rules for free text are based on relations between words and semantic classes of words.

In original WHISK implementation, the learning phase is interleaved with user tagging. However, in our work, documents are tagged before WHISK starts to operate on documents. Therefore tagged instances are not given one by one, but as a whole to the WHISK algorithm. The input given to the WHISK algorithm is shown in Figure 4.1. Under the *AlgorithmMatchSentencesAndEvents*, there are three child elements, *TaggedSentence*, *UnTaggedSentence* and *MatchEvent*. WHISK algorithm uses *TaggedSentence* and *MatchEvent* elements for rule extraction. *UnTaggedSentence* is used for applying the WHISK rules and calculating precision and recall values.

```

<AlgorithmMatchSentencesAndEvents>
  <UnTaggedSentence>
    <Token>90+1</Token>
    <Token>Pareja</Token>
    <Token>{</Token>
    <Token>Anderlecht</Token>
    <Token>}</Token>
    <Token>fouls</Token>
    <Token>.</Token>
  </UnTaggedSentence>
  <TaggedSentence>
    <ExtraMinute>90+1</ExtraMinute>
    <FirstUpperCase>Pareja</FirstUpperCase>
    <Token>{</Token>
    <FirstUpperCase>Anderlecht</FirstUpperCase>
    <Token>}</Token>
    <Token>fouls</Token>
    <Token>.</Token>
  </TaggedSentence>
  <MatchEvent>
    <FoulEvent>
      <Minute>90+1</Minute>
      <PlayerName>Pareja</PlayerName>
      <Team>Anderlecht</Team>
    </FoulEvent>
  </MatchEvent>
</AlgorithmMatchSentencesAndEvents>

```

Figure 4.1: Evaluating Performance of WHISK

4.1.1 Creating hand-tagged training instances

As mentioned in the previous section, WHISK requires pre-processing on textual data. For some domains of structured and semi-structured, textual data is divided into instances that can be HTML tags or some kind of regular expressions. For free text, a syntactic analyzer could process the free text and divide it into clauses, sentences and sentence fragments.

Training instances that refer to the tagged instances are formed by regular expressions. Three types of tags are designated for our corpus: *FirstUpperCase*, *Minute* and *ExtraMinute*. *FirstUpperCase* is the words that start with upper case letters, *Minute* is any kind of number and *ExtraMinute* is number followed with a plus sign and then again followed by another number. These tags of training instances are used by WHISK to create rules. Figure 4.2 shows an example tagged sentence for WHISK algorithm.

```

<TaggedSentence>
  <ExtraMinute>90+1</ExtraMinute>
  <FirstUpperCase>Reina</FirstUpperCase>
  <Token>(</Token>
  <FirstUpperCase>Liverpool</FirstUpperCase>
  <Token>)</Token>
  <Token>makes</Token>
  <Token>a</Token>
  <Token>save</Token>
  <Token>.</Token>
</TaggedSentence>

```

Figure 4.2: An example for tagged instance for WHISK algorithm

4.1.2 Creating a rule from a seed instance

WHISK induces rules top-down that finds the most general rule that covers an instance and extend the rule by adding terms. Top-down rule induction begins with an empty rule, and then adds terms to the rule which covers more instances. After starting empty rule, WHISK grows rule by anchoring the extraction boundaries one slot at a time. WHISK considers two base rules, a rule with terms added just within the extraction boundary (Base1) and a rule with terms added just outside the extraction (Base2). The base rule that covers more instances is selected. As an example, consider the instance given in Figure 4.2. The semantic class *ExtraMinute* matches the first term of slot 1. The terms just outside this slot are the start of the sentence and the following left parenthesis that is the first untagged token after *ExtraMinute*. Base1 skips until it matches with semantic class *ExtraMinute* and Base2 is start of a sentence followed by a left parenthesis. Base1 is selected as anchor since it applies correctly to more instances. WHISK considers two alternatives for extending rule, Base1 uses the *FirstUpperCase* class and Base2 uses left parenthesis. Base1 looks for the *FirstUpperCase* after the

ExtraMinute and Base2 looks for the first left parenthesis after the *ExtraMinute*. Each of these operates correctly on that instance, but Base1 covers more instances so that it is selected as anchor. This process is continued for slot 3 and final anchored rules are created for that instance. Base rules for three slots are shown in Figure 4.3. WHISK's wildcard "*" will skip over tokens until the next literal in the rule.

Anchoring Slot 1:	
Base1:	* <ExtraMinute>
Base2:	* "("
Anchoring Slot 2:	
Base1:	<ExtraMinute> * <FirstUpperCase>
Base2:	<ExtraMinute> * "("
Anchoring Slot 3:	
Base1:	<ExtraMinute> * <FirstUpperCase> * <FirstUpperCase>
Base2:	<ExtraMinute> * <FirstUpperCase> * "(" * ")"

Figure 4.3: Base Rules for anchoring each slot

Generally, base rules are not enough for making correct extractions, in which case more terms are added until the rules covers the seed. WHISK extends rules by adding terms and testing the coverage of that rule in tagged training set. If a term is annotated, it has a user defined semantic class, and WHISK tries to add its semantic class to the rule. In our work, each word, number or punctuation are considered as terms. If the term is not annotated, it is tagged as "Token" and WHISK tries to add to the rule. An example rule that is extracted by WHISK algorithm is illustrated in Figure 4.4.

4.2 Event Extraction with Unique Match Sequence

A Unique Match Sequence is representing similarities and differences between two strings. Before going to details of match sequence, similarity and difference will be explained. A similarity is the representation of similar parts of two strings and a difference is the representation of pair of differing parts between two strings [4].

```

<Rule>
  <Pattern>
    <ExtraMinute/>
    <FirstUpperCase/>
    <Token>{</Token>
    <FirstUpperCase/>
    <Token>}</Token>
    <Token>is</Token>
    <Token>flagged</Token>
    <Token>for</Token>
    <Token>offside</Token>
    <Token>.</Token>
  </Pattern>
  <MatchEvent>
    <Offside>
      <Minute/>
      <PlayerName/>
      <Team/>
    </Offside>
  </MatchEvent>
</Rule>

```

Figure 4.4: Example WHISK Rule

A Unique Match Sequence satisfies the conditions that if a symbol occurs in similarity, it cannot occur in any difference and if a symbol occurs in first unit of difference, it cannot occur in second unit of any difference. These conditions guarantee the uniqueness of match sequence that two strings have either only one unique match sequence or they will not have unique match sequence [4].

The Unique Match Sequence is used for information extraction by finding the similarities and differences in two strings. In Figure 4.5, there are two example strings from our match corpus that Unique Match Sequence is applied. In this example differences between two strings are underlined.

```

['62' , 'Enevoldsen' , '(' , 'AaB' , ')', 'commits' , 'a' , 'foul' ,
'after' , 'challenging' , 'Hassan' , '(' , 'Anderlecht' , ')', '.'] =
event(foulevent, [minute = ['62'] , playername = ['Enevoldsen'] , team =
['AaB'] , playername = ['Hassan'] , team = ['Anderlecht'] ] ).

['88' , 'Frutos' , '(' , 'Anderlecht' , ')', 'commits' , 'a' , 'foul' ,
'after' , 'challenging' , 'Olesen' , '(' , 'AaB' , ')', '.'] =
event(foulevent, [minute = ['88'] , playername = ['Frutos'] , team =
['Anderlecht'] , playername = ['Olesen'] , team = ['AaB'] ] ).

```

Figure 4.5: Two example strings for Unique Match Sequence

After finding differences between two sample strings, their unique match sequence is illustrated in Figure 4.6. The differences are represented within parentheses and the similarities are represented within single quotations.

```
[ (62, 88) (Enevoldsen,Frutos) '(' (AaB,Anderlecht) ')' 'commits' 'a' 'foul'
'after' 'challenging' (Hassan,Olesen) '(' (Anderlecht,AaB) ')' '']
```

Figure 4.6: Unique Match Sequence of two sample strings

After the unique match sequence is found, differences are replaced with variables in an instance of this unique match sequence in order to find a generalized clause. These generalized clauses are used as rules for information extraction. Figure 4.7 shows an example rule, first line represents the generalized clause with variables, the second line represents the extracted information again with variables and the following lines are for the variable conditions that show how many strings they have and the properties of these strings. All of the extracted rules can be found in APPENDIX B. When a sentence is matched with a generalized clause, its variables are replaced with the extracted information.

```
rule([X, Y, '(' , Z, ')', commits, a, foul, after, challenging, W, '(' , T, ')', '.',
event, foulevent, minute, X, playername, Y, team, Z, playername, W, team, T],
[vc(T, [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(W, [vCond(1, [firstUpperCase]), vCond(2, [firstUpperCase, firstUpperCase])]),
vc(Z, [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(Y, [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(X, [vCond(1, [numNum]), vCond(1, [number])])])
```

Figure 4.7: Rule Representation created from Unique Match Sequence

4.3 Discussion

WHISK and Event Extraction with Unique Match Sequence are applied on the same match corpus. Since the data is well formatted, they both have good results. The match corpus has 246 different minute by minute match reports and 26547 events

resides in these match reports. While evaluating the empirical results of WHISK and Event Extraction with Unique Match Sequence, cross-validation is used that ninety percentage of data is used as training instance to form rules and learned rules are applied over ten percentage of data to evaluate the results. Event Extraction with Unique Match Sequence has 1820 correct extraction out of 1828 events. 8 events cannot be extracted due to some reasons such as there is a player name with three words in only one example. Since WHISK can extract rules from one instance, it does not miss these rules. Therefore WHISK has 100 percentage success while extracting events.

Another advantage of WHISK algorithm is that one example of tagged data is enough for finding a rule, but for Event Extraction with Unique Match Sequence, at least two examples are required to extract a rule since it works on the similarities and differences. On the other hand, WHISK works on tagged data, but Event Extraction with Unique Match Sequence does not require tagging the data.

CHAPTER 5

IMPLEMENTATION

The system proposed in this thesis is implemented in Java programming language. As the graphical user interface, Java Swing package of the standard Java software development kit was used. It is a component-based framework and able to fulfill all the graphical interface requirements of the thesis.

In our work, match reports are downloaded from UEFA web site. Crawling and parsing are used in order to create match corpus. NekoHTML is used for parsing the UEFA web site which is a simple HTML scanner that enables parsing the HTML documents and accessing the information using standard XML interfaces. After creating match data, events are extracted from these reports and this event information is mapped into MPEG-7 files for each match.

There is also a user interface for querying the match events. XQuery language is used for searching and querying the MPEG-7 files. XQEngine library is used to process XQuery language over MPEG-7 files. It is full-text search engine for XML documents, utilizing XQuery as its front-end query language.

There are minute based, player name based, team based, match based and event based search options in our system. Users can select one of them or any combination of them. Since there are multiple search options and we are using XQuery, we implemented a module that dynamically generates XQuery according to search options. This module adjusts the necessary joins in an efficient manner.

After the search operation, the user can select one of the results, and if a match video is associated with that match, its video is shown at the top of search results. While playing the video, we use Java Media Player API, a portion of the Java Media Framework (JMF) that enables audio and video within applications.

In Figure 5.1, Corner events that Inter team has taken against Liverpool are shown.

The result has shown the event name, minute of the event, the player who takes the corner and the match result.

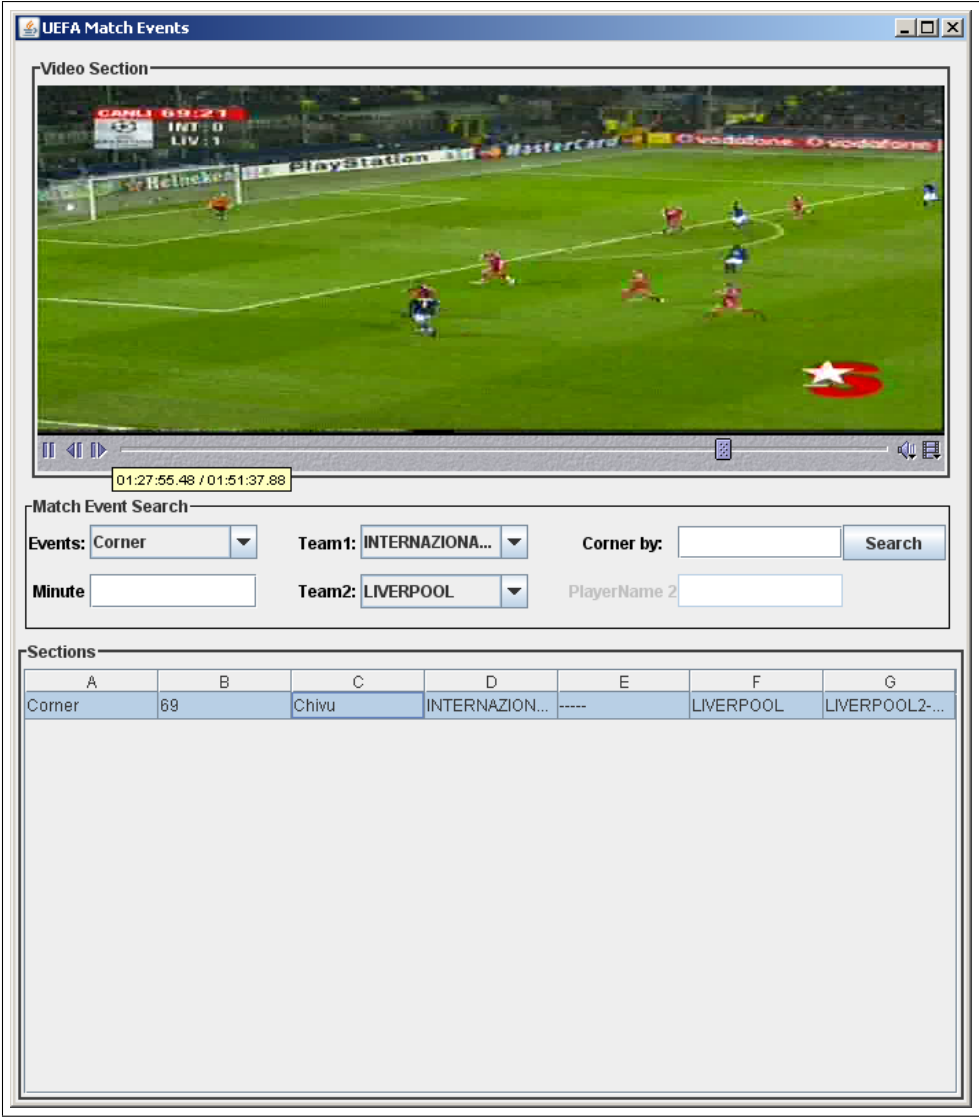


Figure 5.1: Displaying results for event-based querying

Figure 5.2 illustrates querying of all events that player Gerrard was involved in games where Liverpool and Internazionale are opponents. Goal Position at 25 minute is selected and that event is shown at the top of search results.

5.1 Architecture

The general architecture of our system is as follows in Figure 5.3.

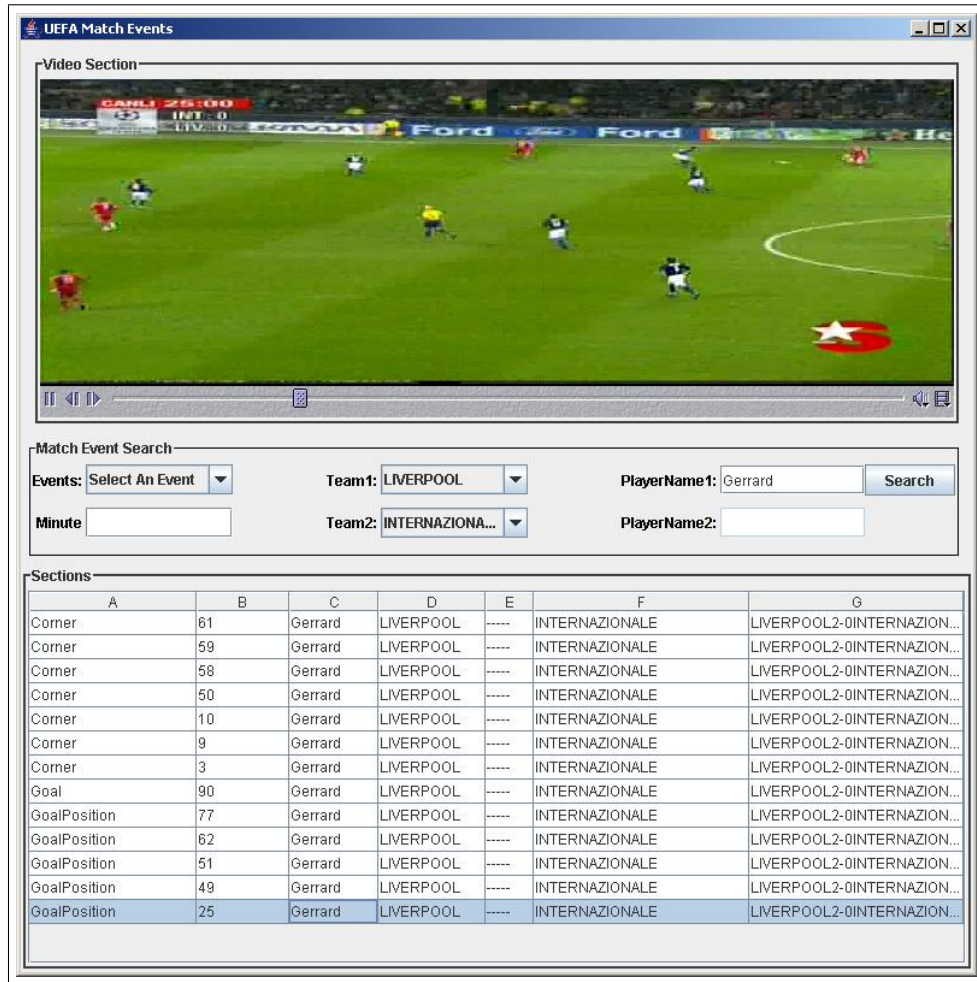


Figure 5.2: Player-based querying

Match reports are provided by Web-Crawler module that downloads the minute by minute match reports. EntityExtractor module tags the sentences according to pre-defined regular expressions. These tagged match reports are used by EventExtractor module for extracting events. EventExtractor module also needs a rule set for event extraction. These rules can be hand written rules or whisk rules that are created by WhiskAlgorithm module. WhiskAlgorithm module creates rules from tagged instances and again these rules can be used for event extraction. MPEG-7 files are created from extracted events by MPEG-7 Generator module.

User interface module provides users a means for match metadata search and query over MPEG-7 match corpus. There are several search options: player based, team based, event based and minute based. According to the search criteria, a XQuery is created and processed over all MPEG-7 files. The results are displayed on the screen

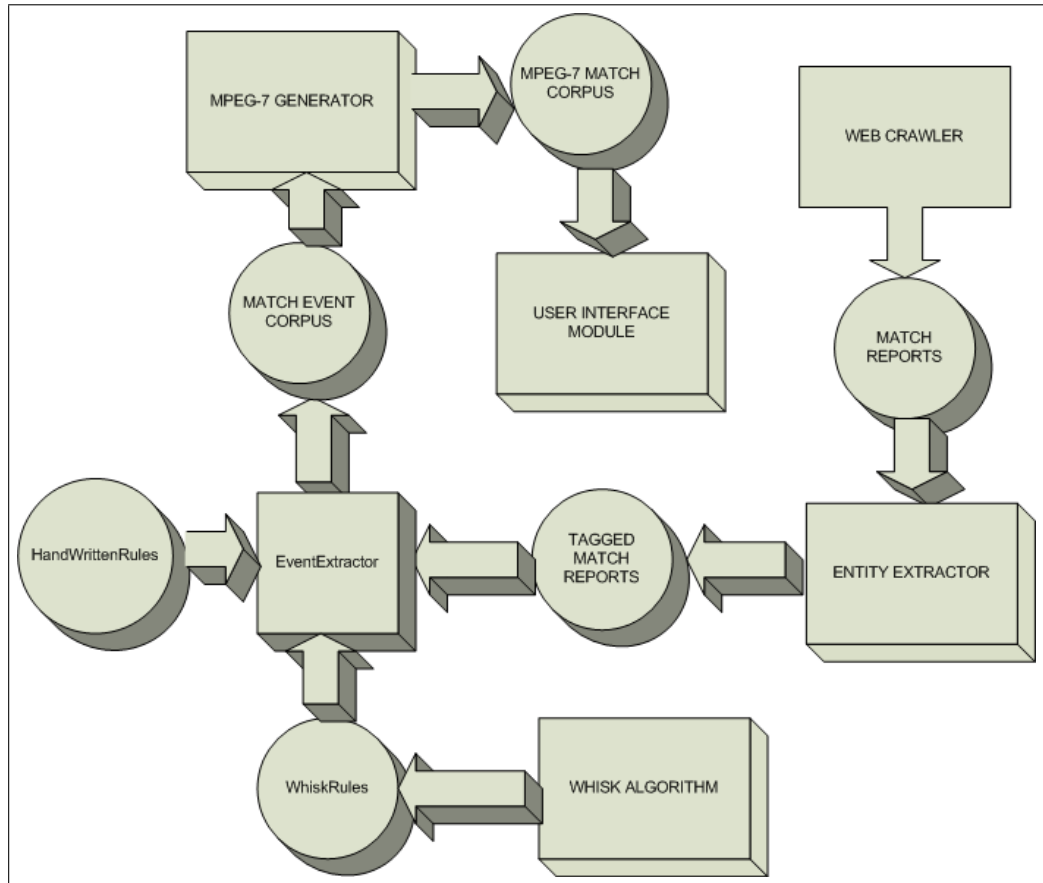


Figure 5.3: The System Architecture

as a table and user has able to watch the event if there is an associated video file for that event. If there is not an associated video file, it gives a warning message.

5.2 System Overview

The user interaction of system is shown in Figure 5.4. The system has search options according to the player name, team name, minute and event name. Any combinations of these fields are also acceptable. A module that generates XQuery is required since XQuery language is used for searching and querying MPEG-7 files. User interface module determines the search options and passes that search options to the XQuery-Generator module. That module prepares the necessary XQuery and evaluates it over the MPEG-7 files.

There are three different types of queries. There are one player events which are performed by one player, and foul event which includes two players from two different teams and substitution event that includes two player from the same team. Figure 5.5

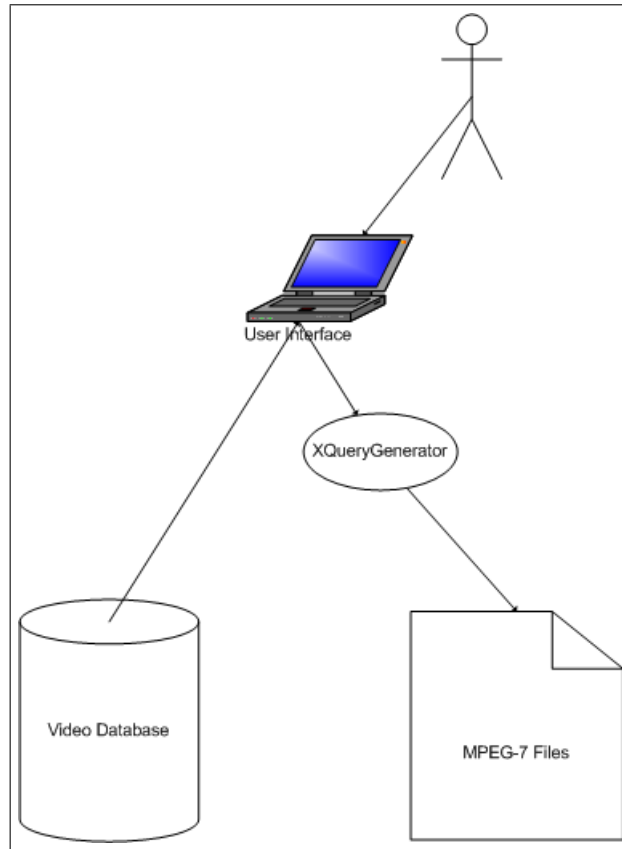


Figure 5.4: System Overview

shows an example of one player event that queries corner event that has been performed by Arsenal. Since two search options, team *Arsenal* and event *Corner* are stored under *Relation* element of MPEG-7; they are joined with *Relation1* and *Relation2*. Since *Corner* event is a one player event, *Foul* and *Substitution* events will be eliminated with *where* condition.

Figure 5.6 illustrates an example query for player *Seedorf* who scores with his team *Milan* against *Shaktar* team. There are four search options and three of these search options are stored under *Relation* element of MPEG-7. Therefore three joins of *Relation* element are required. Again, *Foul* and *Substitution* events are eliminated with *where* condition since *Goal* event is a one player event.

Figure 5.7 illustrates an example query for player *Nesta* who commits a foul with his team *Milan* against *Shaktar* team. Since *Foul* event includes two players from two different teams, its MPEG-7 representation is a little bit different as it is stated in Chapter 3. *Foul* event is composed of two events: *FoulEventCommitted* and *FoulSuffered*. For each of these events, two separate events are created and the relationship

```

<ResultList>
{ for $semanticBase in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase
,$relation1 in $semanticBase/Relation
,$relation2 in $semanticBase/Relation

where
$relation1/@target != "FoulEventCommitted" and $relation1/@target != "FoulSuffered" and
$relation1/@target != "SubstitutionIn" and $relation1/@target != "SubstitutionOut" and
$relation1/@type = "urn:...:agentOf" and $relation2/@target = "ARSENAL"and
$relation2/@type = "urn:...:memberOf" and $semanticBase/Relation/@target = "Corner"

return <Result>
    { $semanticBase/Agent/Name/GivenName }
    { $semanticBase/Relation}
    { $semanticBase/MediaOccurrence/MediaLocator/MediaTime/MediaTimePoint}
  </Result>}
</ResultList>

```

Figure 5.5: XQuery Example for Corner Event

```

<ResultList>{ for $semanticBase in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase
,$relation1 in $semanticBase/Relation
,$relation2 in $semanticBase/Relation
,$relation3 in $semanticBase/Relation

where

$relation1/@target != "FoulEventCommitted" and $relation1/@target != "FoulSuffered" and
$relation1/@target != "SubstitutionIn" and $relation1/@target != "SubstitutionOut" and
$relation1/@type = "urn:...:agentOf" and $semanticBase/Agent/Name/GivenName = "Seedorf" and
$relation2/@target = "MILAN"and $relation2/@type = "urn:...:memberOf" and
$relation3/@target = "SHAKHTAR" and $relation3/@type = "urn:...:hasAccompaniedOf" and
$semanticBase/Relation/@target = "Goal"

return <Result>
    { $semanticBase/Agent/Name/GivenName }
    { $semanticBase/Relation}
    { $semanticBase/MediaOccurrence/MediaLocator/MediaTime/MediaTimePoint}
  </Result>}
</ResultList>

```

Figure 5.6: XQuery Example for Goal Event

between these events is established by another *Foul* event element. Therefore while querying *Foul* event, first *Foul* events are selected and by using its relations, *FoulEventCommitted* and *FoulSuffered* are selected. As it is seen in Figure 5.7, *semanticBase1* element represents the *Foul* event, *semanticBase2* element represents the *FoulEventCommitted* event and *semanticBase3* element represents the *FoulSuffered* event. These elements are joined and their conditional queries of *FoulEventCommitted* and *FoulSuffered* events vary according to the specified search criteria.

Figure 5.8 illustrates an example query for *Substitution* event that is performed with team *Milan* against *Shaktar* team. *Substitution* event is similar to the *Foul* event with its MPEG-7 representation. Therefore same strategy for *Foul* event is applied for


```

<ResultList>
{
for $semanticBase1 in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase
where $semanticBase1/Relation/@target = "FoulEvent"
return
  <FoulEvent>
  {
    for $semanticBase2 in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase,
    $relation2 in $semanticBase2/Relation

    where $semanticBase1/ObjectRef/@idref = $semanticBase2/@id and
    $semanticBase2/Relation/@target = "FoulEventCommitted" and
    $semanticBase2/Agent/Name/GivenName = "Nesta" and
    $relation2/@target = "MILAN" and $relation2/@type = "urn:...:memberOf"
    return
    <PlayerFoul>
    {
      for $semanticBase3 in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase,
      $relation3 in $semanticBase3/Relation

      where $semanticBase1/ObjectRef/@idref = $semanticBase3/@id and
      $semanticBase3/Relation/@target = "FoulSuffered" and
      $relation3/@target = "SHAKHTAR" and $relation3/@type = "urn:...:memberOf"
      return
      <FoulPerson>
      {
        <FoulCommitted>
        {
          {$semanticBase2/Agent/Name/GivenName}
          {$semanticBase2/Relation}
          {$semanticBase2/MediaOccurrence/MediaLocator/MediaTime/MediaTimePoint}
        }
        </FoulCommitted>

        <FoulSuffered>
        {
          {$semanticBase3/Agent/Name/GivenName}
          {$semanticBase3/Relation}
          {$semanticBase3/MediaOccurrence/MediaLocator/MediaTime/MediaTimePoint}
        }
        </FoulSuffered>
      }
    }
  }
}
</FoulEvent>
}
</ResultList>

```

Figure 5.7: XQuery Example for Foul Event

Substitution event. Again *Substitution* event is composed of two separate events that are *SubstitutionIn* and *SubstitutionOut*. The relationship between these events is accomplished by *Substitution* event. Therefore while querying *Substitution* event, first *Substitution* events are selected and then *SubstitutionIn* and *SubstitutionOut* events are joined with *Substitution* event. The conditional queries of *SubstitutionIn* and *SubstitutionOut* events are designated by the search criteria.

5.3 Tools Used for Implementation

5.3.1 NEKOHtml

NEKOHtml is a simple HTML scanner that enables application programmers to parse HTML documents and access the information using standard XML interfaces. NEKO-

```

<ResultList>
{
  for $semanticBase1 in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase
  where $semanticBase1/Relation/@target = "SubstitutionEvent"
  return
  <SubstitutionEvent>
  {
    for $semanticBase2 in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase,
    $relation2 in $semanticBase2/Relation,
    $relation3 in $semanticBase2/Relation

    where $semanticBase1/ObjectRef/@idref = $semanticBase2/@id and
          $semanticBase2/Relation/@target = "SubstitutionIn" and
          $relation2/@target = "MILAN" and
          $relation2/@type = "urn:...:memberOf" and
          $relation3/@target = "SHAKHTAR" and
          $relation3/@type = "urn:...:hasAccompaniedOf"

    return
    <PlayerSubstitution>
    {
      for $semanticBase3 in doc(%MPEG-7_File%)/Mpeg7/Description/Semantics/SemanticBase
      where $semanticBase1/ObjectRef/@idref = $semanticBase3/@id and
            $semanticBase3/Relation/@target = "SubstitutionOut"
      return
      <SubstitutionPerson>

      <SubstitutionIn>
      {
        { $semanticBase2/Agent/Name/GivenName }
        { $semanticBase2/Relation }
        { $semanticBase2/MediaOccurrence/MediaLocator/MediaTime/MediaTimePoint }
      }
      </SubstitutionIn>

      <SubstitutionOut>
      {
        { $semanticBase3/Agent/Name/GivenName }
        { $semanticBase3/Relation }
        { $semanticBase3/MediaOccurrence/MediaLocator/MediaTime/MediaTimePoint }
      }
      </SubstitutionOut>

      </SubstitutionPerson>
    }
    </PlayerSubstitution>
  }
  </SubstitutionEvent>
}
</ResultList>

```

Figure 5.8: XQuery Example for Substitution Event

Html parser can scan HTML files and correct many common mistakes that takes part in HTML documents. It adds automatically missing parent elements, closes end tags and can handle mismatched element tags.

NEKOHtml can be used with existing XNI (Xerces Native Interface) tools without modification or rewriting since it is written using XNI that is a foundation of the Xerces2 implementation. This parser is written in Java and it requires Java 1.3 or higher and Xerces 2.0.0 or higher.

There are some limitations for NEKOHtml usage. These are:

1. NEKOHtml cannot properly generate a well-formed XML document event stream from every HTML document. For instance, some documents have multiple html

tags and XML documents should have a single root element.

2. Code added to the core DOM implementation in Xerces-J 2.0.1 introduced a bug in the HTML DOM implementation based on it.

5.3.2 XQEngine

XQEngine is a full-text search engine for XML documents. It enables integrating XML documents for combinations of keywords such as Google does for HTML documents. However XQuery provides much more powerful search capabilities than HTML-based search engines, because it enables specifying constraints on attributes and element hierarchies with its XPath component.

XQEngine is a component written in Java. It requires a Java programming skill to use since it is not a standalone application. It has a straightforward programming interface that makes it easy to do.

The features of XQEngine can be listed as:

1. It handles multiple documents such as XML files on local hard drive and remote documents on web servers.
2. XQEngine is fast as a pre-indexing type of engine.
3. It has a Java programming API.
4. It is an open source software.
5. XML documents can be indexed.
6. It is an free-text search engine that accommodates XQuery and XPath.

5.3.3 Java Media Framework

The Java Media Framework (JMF) is a Java tool that provides audio, video and other time-based media to be added to Java applications. This tool supports multiple media formats and extends Java Platform, Standard Edition (Java SE) to allow developing multimedia application with Java. JMF handles time-based media, in another saying, media which changes with respect to time.

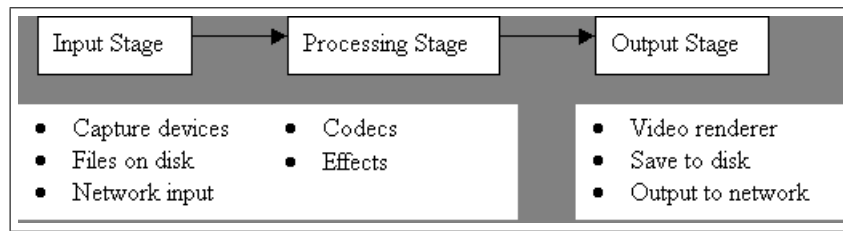


Figure 5.9: Stages of JMF

Stages of Java Media Framework

The JMF architecture is divided into three stages:

In Input Stage, data that can be a capture device such as webcam or TV capture card and a file on disk is read from a source and parsed for the processing stage.

The processing stage will prepare a suitable output by using a number of codecs. These codecs compress or decompress audio and clean up noise.

When the processing step is finished, it passes the stream into the output stage. It will display the local video or transmit it over the network.

Component Architecture

The component architecture of JMF is organized into a number of categories:

- Media Handlers
- Data Sources
- Codecs/Effects
- Renderers
- Mux/Demuxes

JMF should be able to support each type of file with the registered Media Handlers. For the unknown file formats a new Media Handler should be created. A Data Source handler should be able to process streams from different sources. These sources can be network protocols or simple input from disk. Codecs takes an input stream transform it and output it. They have various input and output formats. Renderers resemble codecs but their final output will be the data on screen. Multiplexers and Demultiplexers are for combining multiple streams into a single stream or dividing a single stream into multiple streams. They are useful for saving audio and video data together on a disk.

Presenting Data

JMF provides interfaces for reading, processing and displaying the data. Media data can be integrated into any application such as AWT or Swing with the Player. The processor makes it possible to encode and decode for the Player and adding custom codecs.

CHAPTER 6

CONCLUSION

We present an MPEG-7-based approach to information extraction in soccer domain that targets the automatic generation of MPEG-7 files from match reports. We choose the MPEG-7 standard for our match corpus to make the system more interoperable. Using MPEG-7 is advantageous since our match corpus can be used in any platform for further studies. Also the addition of extra elements to enrich the content is allowed due to the nature of MPEG-7.

A web crawler is used for downloading the match reports from UEFA web site. These match reports are annotated using regular expressions. According to the hand-written rule set, match events are extracted and converted to the MPEG-7 standard. The events extracted by using hand-written have been used for computing the precision and recall of information extraction algorithms [34].

In addition, two information extraction algorithms are used instead of hand-written rules for extracting events from free text match reports. Whisk and Unique match sequence techniques are applied on our match corpus. Both of them results nearly one hundred percent precision and recall since the data set is well formed.

Finally, we adapted an interface for querying match events over MPEG-7 files by using XQuery Language. However since there are several kinds of query types we need a module that dynamically prepares XQueries according to search criteria. These queries are processed over all of the matches and results are displayed in our interface. If user selects a result and there is an associated match video, it will be shown in our interface.

There are some handicaps for synchronizing the match video and the information extracted from minute by minute match reports. Sometimes, the information in the text will not be appropriate with the real video. Therefore, different techniques could

be used for mapping text to corresponding scenes in video. Analyzing video scenes or image processing techniques could be used [22].

On the other hand, since our data set is well formed, information extraction algorithms have approximately one hundred percentage precision and recall. Data set could be changed and performance of algorithms could be tried in another data set that is closer to free text.

Again, since our data set is well formed, it is possible to annotate the match reports with regular expressions. However this will be more challenging problem if the text is not well formed. An annotation tool could be provided and it could output the annotated match in the same way that our system did with regular expressions.

REFERENCES

- [1] Peter van Beek, Ana B. Benitez, Joerg Heuer, Jose Martinez, Philippe Salembier, John Smith, Toby Walker, M6155 MPEG-7 Multimedia Description Schemes XM (Version 3.1) , July 2000, Beijing
- [2] Benitez, A. B., Rising, H., Jørgensen, C., Leonardi, R., Bugatti, A., Hasida, K., Mehrotra, R., Tekalp, A. M., Ekin, A., Walker, T., Semantics of Multimedia in MPEG-7, Proceedings of 2002 IEEE Conference on Image Processing (ICIP-2002), Rochester, New York, USA, 22-25 September 2002.
- [3] P. Buitelaar et al. Generating and visualizing a soccer knowledge base. In Proc. of the EACL'06 Demo Session, Trento, Italy, 2006.
- [4] Ilyas Cicekli, and Nihan Kesim Cicekli, Generalizing Predicates with String Arguments, Applied Intelligence, Vol. 25, No. 1, (2006), pp: 23-36.
- [5] Leszek Cieplinski, Whoi-Yul Kim, Jens-Rainer Ohm, Mark Pickering. Multimedia content description interface - Part 3 Visual, 7 2001.
- [6] Clark, J., DeRose, S., XML Path Language (XPath) Version 1.0 W3C Recommendation, <http://www.w3.org/TR/xpath>, Last Update Date: 16 November 1999
- [7] C. Dolbear and M. Brady. Soccer highlights generation using a-priori semantic knowledge. In Proc. IEEE conference on Visual Information Engineering, VIE2003, University of Surrey, UK, July 2003.
- [8] Freitag, D. (1998). Information extraction from HTML: Application of a general learning approach. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), pp. 517-523, Madison, WI. AAAI Press / The MIT Press.
- [9] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In Proc. 17th Nat. Conf. Artificial Intelligence, 2000.
- [10] Andrew Graves and Mounia Lalmas. Video retrieval using an MPEG-7 based inference network. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pages 339-346. ACM Press, 2002.
- [11] Ziya Özkan Göktürk, Nihan Kesim Çiçekli, İlyas Çiçekli. Metadata Extraction from Text in Soccer Domain. ISCIS 2008, 23th International Symposium, Istanbul, Turkey, October 27-29, 2008
- [12] Ralph Grishman. Information extraction: Techniques and challenges. In Information Extraction (International Summer School SCIE-97). Springer-Verlag, 1997.

- [13] Jason Hunter, Xquery.com: Specifications, Articles, Mailing List, and Vendors, <http://www.xquery.com/>, Last Update Date: 2003
- [14] Nemrava, Jan Buitelaar, Paul Svatek, Vojtich Declerck, Thierry. Event Alignment For Cross-Media Feature Extraction In The Football Domain. In: WIAMISS'07, Santorini, 2007: IEEE Computer Society, pages 1-3
- [15] Werner Klieber, Using MPEG-7 for Multimedia retrieval. Graz University of Technology, 2003
- [16] Obasanjo D., An Exploration of XML in Database Management Systems, <http://www.25hoursaday.com/StoringAndQueryingXML.html>, Last Update Date: 2001
- [17] Atul Puri and Alexandros Eleftheriadis. MPEG-4: an object-based Multimedia coding standard supporting mobile applications. *Mobile Networks and Applications*, 3(1):5-32, 1998.
- [18] Stephen Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning* 34(1-3): 233-272 (1999)
- [19] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.
- [20] Steegmans, B., XML for DB2 Information Integration, ibm-redbooks, July 2004
- [21] Changsheng Xu, Jinjun Wang and Yifan Zhang, "A Novel Framework for Semantic Annotation and Personalized Retrieval of Sports Video". Vol 10; Numb 3, pages 421-436. *IEEE Transaction on Multimedia*, 2008
- [22] C. Xu, J. Wang, K. Wan, Y. Li, and L. Duan, "Live sports event detection based on broadcast video and web-casting text" in *Proc. ACM Multimedia*, 2006, pp. 221-230.
- [23] Yankova, M., S. Boytcheva (2003) "Focusing on Scenario Recognition in Information Extraction", In *Proc. EACL-2003*, pp.41-48.
- [24] Bourret R., XML Database Products: Native XML Databases, <http://www.rpbourret.com/xml/XMLDatabaseProds.htm>, Last Access Date: December 2007
- [25] Bourret R., <http://www.rpbourret.com/xml/XMLAndDatabases.htm>, Last Access Date: February 2008
- [26] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Extensible Markup Language (XML) 1.0 W3C Recommendation, <http://www.w3.org/TR/1998/REC-xml-19980210>, Last Access Date: February 2008
- [27] MPEG-21 Overview. <http://mpeg.telecomitalialab.com/standards/mpeg-21/mpeg-21.htm>, Last Access Date: March 2008.
- [28] MPEG-4 Systems MPEG-J. <http://mpeg.telecomitalialab.com/faq/mp4-sys/sys-faq-mpegj.htm>, Last Access Date: April 2008.
- [29] Description Definition Language (DDL) Home Page. <http://archive.dstc.edu.au/mpeg7-ddl>, Last Access Date: June 2008.

- [30] MPEG-7 Overview. International Organization for Standardization. <http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm>. Last Access Date: July 2008.
- [31] The MPEG Homepage. <http://mpeg.telecomitalialab.com/>, Last Access Date: June 2008.
- [32] World Wide Web Consortium Issues XML Schema as a Candidate Recommendation. <http://www.w3.org/2000/10/xml-schema-pressrelease.html>, Last Access Date: June 2008.
- [33] MPEG-7 Overview. <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm>, Last Access Date: October 2007
- [34] The SmartWeb Project. http://www.smartweb-project.org/start_en.html, Last Access Date: August 2008

APPENDIX A

RULE SET FOR MATCH EVENTS

```

<RuleSet>
  <Rule>
    <Pattern>
      <Minute></Minute>
      <PlayerName></PlayerName>
      <Token>(</Token>
      <Team></Team>
      <Token>)</Token>
      <Token>is</Token>
      <Token>cautioned</Token>
      <Token>by</Token>
      <Token>the</Token>
      <Token>referee</Token>
      <Token>.</Token>
    </Pattern>
    <MatchEvent>
      <CautionedEvent>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Team></Team>
      </CautionedEvent>
    </MatchEvent>
  </Rule>
  <Rule>
    <Pattern>
      <Minute></Minute>
      <PlayerName></PlayerName>
      <Token>(</Token>
      <Team></Team>
      <Token>)</Token>
      <Token>delivers</Token>
      <Token>the</Token>
      <Token>corner</Token>
      <Token>.</Token>
    </Pattern>
    <MatchEvent>
      <CornerEvent>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Team></Team>
      </CornerEvent>
    </MatchEvent>
  </Rule>

```

```

<Rule>
  <Pattern>
    <Minute></Minute>
    <PlayerName></PlayerName>
    <Token>(</Token>
    <Team></Team>
    <Token>)</Token>
    <Token>takes</Token>
    <Token>the</Token>
    <Token>corner</Token>
    <Token>.</Token>
  </Pattern>
  <MatchEvent>
    <CornerEvent>
      <Minute></Minute>
      <PlayerName></PlayerName>
      <Team></Team>
    </CornerEvent>
  </MatchEvent>
</Rule>
<Rule>
  <Pattern>
    <Minute></Minute>
    <PlayerName></PlayerName>
    <Token>(</Token>
    <Team></Team>
    <Token>)</Token>
    <Token>gives</Token>
    <Token>away</Token>
    <Token>a</Token>
    <Token>free-kick</Token>
    <Token>following</Token>
    <Token>a</Token>
    <Token>challenge</Token>
    <Token>on</Token>
    <PlayerName></PlayerName>
    <Token>(</Token>
    <Team></Team>
    <Token>)</Token>
    <Token>.</Token>
  </Pattern>
  <MatchEvent>
    <FoulEvent>
      <Minute></Minute>
      <PlayerName></PlayerName>
      <Team></Team>
      <PlayerName></PlayerName>
      <Team></Team>
    </FoulEvent>
  </MatchEvent>
</Rule>
<Rule>
  <Pattern>
    <Minute></Minute>
    <PlayerName></PlayerName>
    <Token>(</Token>
    <Team></Team>

```

```

        <Token>)</Token>
        <Token>fouls</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <FoulEvent>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </FoulEvent>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>commits</Token>
        <Token>a</Token>
        <Token>foul</Token>
        <Token>after</Token>
        <Token>challenging</Token>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <FoulEvent>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
            <PlayerName></PlayerName>
            <Team></Team>
        </FoulEvent>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>delivers</Token>
        <Token>the</Token>
        <Token>free-kick</Token>
        <Token>on</Token>
        <Token>target</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <Free-Kick>
            <Minute></Minute>

```

```

                                <PlayerName></PlayerName>
                                <Team></Team>
                                </Free-Kick>
                            </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>delivers</Token>
        <Token>the</Token>
        <Token>free-kick</Token>
        <Token>wide</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <Free-Kick>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </Free-Kick>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>strikes</Token>
        <Token>a</Token>
        <Token>free</Token>
        <Token>kick</Token>
        <Token>into</Token>
        <Token>the</Token>
        <Token>wall</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <Free-Kick>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </Free-Kick>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>

```

```

        <Token>hits</Token>
        <Token>the</Token>
        <Token>crossbar</Token>
        <Token>on</Token>
        <Token>free-kick</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <Free-Kick>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </Free-Kick>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <Score></Score>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>scores!</Token>
    </Pattern>
    <MatchEvent>
        <Goal>
            <Minute></Minute>
            <Score></Score>
            <PlayerName></PlayerName>
            <Team></Team>
        </Goal>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <Score></Score>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>scores</Token>
        <Token>an</Token>
        <Token>own</Token>
        <Token>goal</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <OwnGoal>
            <Minute></Minute>
            <Score></Score>
            <PlayerName></PlayerName>
            <Team></Team>
        </OwnGoal>
    </MatchEvent>
</Rule>

```

```

<Rule>
  <Pattern>
    <Minute></Minute>
    <Score></Score>
    <PlayerName></PlayerName>
    <Token>(</Token>
    <Team></Team>
    <Token>)</Token>
    <Token>scores</Token>
    <Token>on</Token>
    <Token>free-kick</Token>
    <Token>!</Token>
  </Pattern>
  <MatchEvent>
    <GoalFreeKick>
      <Minute></Minute>
      <Score></Score>
      <PlayerName></PlayerName>
      <Team></Team>
    </GoalFreeKick>
  </MatchEvent>
</Rule>
<Rule>
  <Pattern>
    <Minute></Minute>
    <PlayerName></PlayerName>
    <Token>(</Token>
    <Team></Team>
    <Token>)</Token>
    <Token>misses</Token>
    <Token>the</Token>
    <Token>target</Token>
    <Token>.</Token>
  </Pattern>
  <MatchEvent>
    <GoalPosition>
      <Minute></Minute>
      <PlayerName></PlayerName>
      <Team></Team>
    </GoalPosition>
  </MatchEvent>
</Rule>
<Rule>
  <Pattern>
    <Minute></Minute>
    <PlayerName></PlayerName>
    <Token>(</Token>
    <Team></Team>
    <Token>)</Token>
    <Token>has</Token>
    <Token>an</Token>
    <Token>effort</Token>
    <Token>on</Token>
    <Token>goal</Token>
    <Token>.</Token>
  </Pattern>
  <MatchEvent>

```



```

        <GoalPosition>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </GoalPosition>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>has</Token>
        <Token>a</Token>
        <Token>shot</Token>
        <Token>blocked</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <GoalPosition>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </GoalPosition>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>handles</Token>
        <Token>the</Token>
        <Token>ball</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <GoalPosition>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </GoalPosition>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>is</Token>
        <Token>flagged</Token>
    </Pattern>

```

```

        <Token>for</Token>
        <Token>offside</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <Offside>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </Offside>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>is</Token>
        <Token>adjudged</Token>
        <Token>to</Token>
        <Token>be</Token>
        <Token>in</Token>
        <Token>an</Token>
        <Token>offside</Token>
        <Token>position</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <Offside>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </Offside>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <Token>The</Token>
        <Token>ball</Token>
        <Token>is</Token>
        <Token>handled</Token>
        <Token>in</Token>
        <Token>the</Token>
        <Token>penalty</Token>
        <Token>area</Token>
        <Token>by</Token>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>leading</Token>
        <Token>to</Token>
        <Token>a</Token>
        <Token>penalty</Token>

```

```

        </Pattern>
        <MatchEvent>
            <PenaltyEvent>
                <Minute></Minute>
                <PlayerName></PlayerName>
                <Team></Team>
            </PenaltyEvent>
        </MatchEvent>
    </Rule>
    <Rule>
        <Pattern>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Token>(</Token>
            <Team></Team>
            <Token>)</Token>
            <Token>missed</Token>
            <Token>the</Token>
            <Token>penalty</Token>
            <Token>saved</Token>
            <Token>by</Token>
            <Token>the</Token>
            <Token>goalkeeper</Token>
            <Token>.</Token>
        </Pattern>
        <MatchEvent>
            <PenaltyMiss>
                <Minute></Minute>
                <PlayerName></PlayerName>
                <Team></Team>
            </PenaltyMiss>
        </MatchEvent>
    </Rule>
    <Rule>
        <Pattern>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Token>(</Token>
            <Team></Team>
            <Token>)</Token>
            <Token>is</Token>
            <Token>dismissed</Token>
            <Token>.</Token>
        </Pattern>
        <MatchEvent>
            <RedCard>
                <Minute></Minute>
                <PlayerName></PlayerName>
                <Team></Team>
            </RedCard>
        </MatchEvent>
    </Rule>
    <Rule>
        <Pattern>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Token>(</Token>

```

```

        <Team></Team>
        <Token></Token>
        <Token>makes</Token>
        <Token>a</Token>
        <Token>save</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <SaveGoal>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <Team></Team>
        </SaveGoal>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Token>in</Token>
        <Token>)</Token>
        <Token>-</Token>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Token>out</Token>
        <Token>)</Token>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
    </Pattern>
    <MatchEvent>
        <Substition>
            <Minute></Minute>
            <PlayerName></PlayerName>
            <PlayerName></PlayerName>
            <Team></Team>
        </Substition>
    </MatchEvent>
</Rule>
<Rule>
    <Pattern>
        <Minute></Minute>
        <PlayerName></PlayerName>
        <Token>(</Token>
        <Team></Team>
        <Token>)</Token>
        <Token>is</Token>
        <Token>shown</Token>
        <Token>a</Token>
        <Token>yellow</Token>
        <Token>card</Token>
        <Token>.</Token>
    </Pattern>
    <MatchEvent>
        <YellowCard>
            <Minute></Minute>

```

```

                                <PlayerName></PlayerName>
                                <Team></Team>
                                </YellowCard>
                            </MatchEvent>
                        </Rule>
                    <Rule>
                        <Pattern>
                            <Minute></Minute>
                            <PlayerName></PlayerName>
                            <Token>(</Token>
                            <Team></Team>
                            <Token>)</Token>
                            <Token>is</Token>
                            <Token>booked</Token>
                            <Token>.</Token>
                        </Pattern>
                        <MatchEvent>
                            <YellowCard>
                                <Minute></Minute>
                                <PlayerName></PlayerName>
                                <Team></Team>
                            </YellowCard>
                        </MatchEvent>
                    </Rule>
                </RuleSet>

```

APPENDIX B

EXTRACTED RULES FOR EVENT EXTRACTION WITH UNIQUE MATCH SEQUENCE

```
rule([v(1), v(2), '(', v(3), ')', commits, a, foul, after, challenging, v(4), '(', v(5), ')', ')',
event, foulevent, minute, v(1), playername, v(2), team, v(3), playername, v(4), team, v(5)],
[vc(v(5), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(4), [vCond(1, [firstUpperCase]), vCond(2, [firstUpperCase, firstUpperCase])]),
vc(v(3), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])
```

```
rule([v(1), v(2), '(', v(3), ')', gives, away, a, 'free-kick', following, a, challenge, on, v(4), '(',
v(5), ')', ')',
event, foulevent, minute, v(1), playername, v(2), team, v(3), playername, v(4), team, v(5)],
[vc(v(5), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(4), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(3), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])
```

```
rule([v(1), v(2), '(', v(3), ')', fouls, ')',
event, foulevent, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
Case, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])
```

```
rule([v(1), v(2), '(', v(3), ')', delivers, the, corner, ')',
event, cornerevent, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUp-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [firstUpperCase]), vCond(2, [firstUpperCase, firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])
```

```
rule([v(1), v(2), '(', v(3), ')', takes, the, corner, ')',
event, cornerevent, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUp-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [firstUpperCase]), vCond(2, [firstUpperCase, firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])
```

```
rule([v(1), v(2), '(', v(3), ')', makes, a, save, ')',
event, savegoal, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
Case, firstUpperCase]), vCond(1, [firstUpperCase])]),
```

vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', handles, the, ball, '.',
event, goalposition, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', has, a, shot, blocked, '.',
event, goalposition, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', has, an, effort, on, goal, '.',
event, goalposition, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', misses, the, target, '.',
event, goalposition, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), v(3), '(', v(4), ')', 'scores!'
event, goal, minute, v(1), score, v(2), playername, v(3), team, v(4)], [vc(v(4), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(3), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [numNum])]),
vc(v(1), [vCond(2, [number, 'Ex']), vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', in, ')', -, v(3), '(', out, ')', '(', v(4), ')',
event, substitution, minute, v(1), playername, v(2), playername, v(3), team, v(4)], [vc(v(4), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(3), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', is, flagged, for, offside, '.',
event, offside, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', is, adjudged, to, be, in, an, offside, position, '.',
event, offside, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', hits, the, crossbar, on, 'free-kick', '.',
event, free-kick, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [firstUpperCase]), vCond(2, [firstUpperCase, firstUpperCase])]),

```

vc(v(1), [vCond(1, [numNum]), vCond(1, [number])]))

rule([v(1), v(2), '(', v(3), ')', strikes, a, free, kick, into, the, wall, '.',
event, free-kick, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
Case, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', delivers, the, 'free-kick', on, target, '.',
event, free-kick, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
Case, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', delivers, the, 'free-kick', wide, '.',
event, free-kick, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(1, [firstUpper-
Case]), vCond(2, [firstUpperCase, firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(2, [number, 'Ex']), vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', is, booked, '.',
event, yellowcard, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUp-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', is, shown, a, yellow, card, '.',
event, yellowcard, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUp-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [firstUpperCase]), vCond(2, [firstUpperCase, firstUpperCase])]),
vc(v(1), [vCond(2, [number, 'Ex']), vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', is, cautioned, by, the, referee, '.',
event, cautionedevent, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUp-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(2, [number, 'Ex']), vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', is, dismissed, '.',
event, redcard, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUpper-
Case, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), v(3), '(', v(4), ')', scores, on, 'free-kick!',
event, goalfreekick, minute, v(1), score, v(2), playername, v(3), team, v(4)], [vc(v(4), [vCond(2,
[firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(3), [vCond(2, [firstUpperCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [numNum])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

rule([v(1), v(2), '(', v(3), ')', missed, the, penalty, saved, by, the, goalkeeper, '.',
event, penaltymiss, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(1, [firstUp-
perCase])]),
vc(v(2), [vCond(1, [firstUpperCase]), vCond(2, [firstUpperCase, firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

```



```

rule([v(1), v(2), v(3), '(', v(4), ')', scores, an, own, goal, '.',
event, owngoal, minute, v(1), score, v(2), playername, v(3), team, v(4)], [vc(v(4), [vCond(1,
[firstUpperCase])]),
vc(v(3), [vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [numnum])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

```

```

rule([v(1), 'The', ball, is, handled, in, the, penalty, area, by, v(2), '(', v(3), ')', leading, to, a,
penalty,
event, penaltyevent, minute, v(1), playername, v(2), team, v(3)], [vc(v(3), [vCond(2, [firstUp-
perCase, firstUpperCase]), vCond(1, [firstUpperCase])]),
vc(v(2), [vCond(1, [firstUpperCase])]),
vc(v(1), [vCond(1, [numNum]), vCond(1, [number])])])

```