CHAPTER 7

SEMANTIC INFORMATION EXTRACTION BASED IR

7.1 INTRODUCTION

All the approaches discussed so far consider all documents to be represented based on the words present in it. These approaches attempted to semantically enhance the words through various methods. Another perspective is to consider a limited amount of semantic processing of the document in order to obtain a logical methodology of important components of the document. The representation of a document using a logical representation allows information available in the document to be exposed for inferencing. However due to the presence of a large number of documents in the internet and the inadequacy of sophisticated NLP technology, there is a need to logically represent only important components of the document utilizing goal directed heuristic approaches. Information extraction can be viewed as a goal directed search for important pieces of information from a document.

In this work a limited logical representation or knowledge representation model of the document has been constructed based on content words and specially formed templates pertaining to the domain of the document under consideration. The overall domain concept hierarchy is semi-automatically built with semantic relations and the content words of the glosses (Manjula et al 2002a). Frequently occurring words and their
semantically related words (available from WordNet) are used to automatically create a document schema or template. The template of a document denotes a set of nodes in the domain concept tree. The conceptual related words (available from WordNet glosses) are used as arguments for the template or schemas. The information present in the schema is then represented using a knowledge representation technique. The aim of this work is to extract concepts/entities by using thematic role and heuristic rules explained in the previous chapter, to fill in or initiate the templates. These templates are represented as predicates when predicate logic is used as the logical representation mechanism and as terminological facts when Description Logic is used as the representation mechanism. This type of logical representation of chosen concepts of the documents allows inferencing about the overall content of the document. Thus the semantic information extraction from the document has been utilized for improving the IR performance. The semantic information extraction from the document has been utilized to improve the performance of IR.

7.2 GENERATION OF DOMAIN CONCEPT TREE

The primary task in creating a knowledge representation model is the creation of a conceptual taxonomy for the domain under consideration. In this work, a number of documents in the domain under consideration are utilized for manually constructing an initial domain concept tree. This tree is then incrementally enhanced using concepts associated with words of the initial tree. This is done using the WordNet taxonomy for the domain. Using the gloss part of the WordNet, the concepts or predicates present in the conceptual taxonomy
are extracted. The generation of hierarchical conceptual taxonomy has been performed using semantic based text mining techniques (Manjula et al 2002a).

The domain concept trees are a hierarchical representation of concepts whose relations appear explicitly or implicitly in the WordNet. The most important relation is, of course, the IS-A one, which allows one to build taxonomy of concepts related by the hyponym-hypernym links. The domain concept tree is a hierarchy that contains only concepts that are of interest of the user. Its structure defines the generalizations and partitions that the user wants to make while summarizing and analyzing the data. For a given domain, the first step is the construction of the conceptual taxonomy. The defining features of its concepts provide the context for a concept. The word defining features are taken from the gloss of the WordNet and the semantic connections of the concepts from relations available in the WordNet. When a concept belongs to a hierarchy, it inherits the properties of its Hypernym. A concept inherits properties through some of its defining, feature concepts. The idea is to transform each synset's gloss into a defining feature directed tree with synsets as nodes and lexical relations as links. Finally the knowledge base for the particular domain is represented as interlinked domain trees. The synsets at the nodes of the domain concept tree are used as predicates or facts in the representation.

The conceptual taxonomy obtained for the tourism domain is shown in Figure 7.1. The next step is the selection of a template for the new document under consideration utilizing frequently occurring words of the document. Once the template is selected, the associated arguments have to be determined.
The first task is the determination of the correct sense of the template word and its associated arguments. The correct sense of the word associated with the template described in the previous section is determined by applying the concept based word sense disambiguation algorithm described in chapter 4. The associated arguments for a given template are also automatically extracted from the gloss of the synsets using WordNet.

Example:

Temple is taken as name of the Template
The gloss taken from the WordNet for temple with content words are underlined.

An edifice for worship of a deity

The synonym for edifice is place and the synonym for deity is god. The synonyms are used to select appropriate words for arguments that can later be initiated by heuristic rules.

Hence the associated arguments for temple are name, place and god.

In order to fit the argumented template associated with the document into a domain to which the document belongs, it is necessary to fit the template into an appropriate domain oriented concept tree. Thus, before starting the knowledge extraction process for the documents of a domain, a conceptual taxonomy with hierarchical organization of templates and associated arguments are created and made available.

7.4 PREDICATE LOGIC REPRESENTATION

The uninstantiated templates obtained from the previous step are used for obtaining a logical representation. In one approach, predicate logic has been selected as the representational mechanism. The relational schema storage methodology can be mapped with the predicates. For this representation, the templates, which are components of the conceptual taxonomy, are represented using the predicate-argument structure as shown below:
where $P_{11;12;13\ldots1\ N_k}$ are synonyms predicates and each associated argument is represented as synonyms and $N_k$ denotes the number of synonyms, $M$ denotes the number of arguments, $q$ denotes the number of predicates, and ‘,’ and ‘;’ operators denote the AND and OR operators.

The uninstantiated arguments of the predicates are mapped to fields in the database. Heuristic rules discussed in the previous chapter have been used to instantiate the arguments and populate the database. The database representation allows the use of semantic query optimization techniques in the information retrieval process. The semantic information needed for the optimization is given in the form of rules and integrity constraints. In addition, semantic information can be used in the query-processing phase in order to obtain the relevant results in an efficient way. The information extracted and stored in databases is queried effectively using semantic query optimization techniques.

### 7.4.1 Semantics in IR Query Optimization

Our approach to query optimization is the use of semantic knowledge to transform a query into another form that can be executed in a more efficient manner but still yielding the same result as the original query. Two queries are semantically equivalent if they return identical answers from a database state.
that is consistent with the semantic knowledge. Usually, this semantic
knowledge is in the form of rules which are generated either during the query
process itself or are constructed according to predefined heuristics. Two types
of rules namely semantic rules and heuristics rules are used in this work.
Semantic rules are domain specific and are manually defined so that they can
be used to optimize a number of semantically similar queries. In addition to
semantic rules, certain integrity constraints have also been specified in order to
avoid unnecessary records of the database of extracted information from being
processed. Heuristic rules used to extract relations between content words in the
extraction phase are also made available to the query processing phase so as to
enable relations to be defined between content words extracted from the query.

Thus Semantic Query Optimization (SQO) uses semantic information
derived from the data itself, to generate a set of alternative yet equivalent
queries according to the given transformation rules. When the input query is
given, it is fully disambiguated using the concept based word sense
disambiguation algorithm described in chapter 4. The disambiguated query is
then converted to the specific format, which incorporates the semantic
information obtained using the relations, and semantic and heuristic rules
available from the extraction module.

The first phase of the query optimisation is the preprocessing of the
natural language query given by the user. The query is then disambiguated
using the concept based WSD. In the next step, relations between the content
words of the query are determined using heuristic rules and specific types of
relations are defined for the particular domain. In the next phase semantic rules
and integrity constraints are used to transform the query into a set of
semantically equivalent queries which can possibly be evaluated more efficiently. Semantic rules are used to reduce the number of attributes that need to be incorporated into the query, that is, in essence, it reduces the vertical space of the query. Integrity constraints on the other hand, eliminate records of the database that do not satisfy the constraint and in essence reduce the horizontal space of the query.

The tourist domain is taken as an example. A set of queries given by the user is converted into semantically equivalent queries by the heuristic rules and semantic rules. This section explains how the query is processed. The few semantic rules used are:

1. All star hotels have restaurant and parking facilities.
2. If there exists a temple there exists a festival month every year.
3. All tourist spots have hotels.
4. All Shiva temples have nandhi.
5. For all beach resorts there exists a sea.
6. The height of Rajagopuram is at least 50 feet high.
7. If a park has water games it is built after 1990.
8. For all hill stations their altitude is high.
9. If there is a festival there is an exhibition show.
10. All temples have devasthanams.
Consider the following query from the tourist database

Input Query

Query 1:

Find all the Shiva temples with Nandhi in Chennai with area >2000 sq feet.
The system reformulates the query into new query.

Query 2:

Find all Shiva temples in Chennai with area >2000 sq feet.

The optimized query is equivalent to the original query since it is derived from the semantic rule -All Shiva temples have Nandhi.

The relations between the content words of the query are then determined using heuristic rules and enable the system to provide the needed information.

Relations Formed:

All Shiva temples with nandhi in Chennai with area >2000 sq feet.
Integrity Constraint (IC)

Integrity constraints state the range of the given database attribute. For numeral attributes, the IC shows the minimal and maximal value. String type attributes are not used in this system.

The defined IC of this system is:

- > greater than
- < less than
- = equal to

for the above example, the IC is area >2000 sq feet and the arguments of the template, Temple are name, area, location.

area >2000 sq feet: place (name, area, location)

The system searches for attributes name=shiva, location=chennai, area >2000 sq feet, the corresponding names for the above attributes are detected using the rules and the relations and the result displayed is the names of Shiva temples in Chennai with area >2000 sq feet.

Once the list of the documents meeting the relations is determined, the complete text matching the relations is declared as relevant information and presented to the user.
7.4.2 Performance Evaluation

The performance analysis of the query optimization has been studied by considering the tourist domain. Graphs are plotted with the number of rules fired for each query before and after optimization. The time for execution of the query is noted down before and after optimization. The time for execution is taken along the Y-axis and the corresponding query number in the database along the X-axis. The difference in the time can be viewed as the reduction in search space due to the addition of constraints to the database. The performance is analyzed by considering the samples below.

Query 1

Name all the star hotels with restaurant and parking facilities in Trichy.

With optimization

The star hotels with restaurant and parking facilities in Trichy.

Rule 1

'place'

Optimised query

All star hotels in Trichy.

The number of rules fired = 1
Without Optimisation

The star hotels with restaurant and parking facilities in Trichy.

\[ \text{`Coo`} \quad \text{Conj'} \quad \text{`place`} \]

The number of rules fired = 3

Query 2

Find all the Shiva temples with Nandhi in Chennai with area >2000 sq feet.

With optimization

All Shiva temples with nandhi in Chennai with area >2000 sq feet.

\[ \text{Rule 4} \quad \text{Place} \quad \text{Att} \quad \text{IC} \]

Optimized query : Find all Shiva temples in Chennai with area >2000 sq feet.

The number of rules fired = 3.
Without optimization

All Shiva temples with nandhi in Chennai with area >2000 sq feet.

The number of rules fired = 4

Query 3

List all the beach resorts near sea in Mahabalipuram

With optimization

All the beach resorts near sea in Mahabalipuram.

Optimised query: List all beach resorts in Mahabalipuram.
No of rules fired: 1

Without optimization

All the beach resorts near sea in Mahabalipuram

No of rules fired: 2
Query 4

Name the hill station with high altitude around Madurai.

With optimization

Name the hill station with high altitude around Madurai.

Optimized query Name the hill station around Madurai.
No of rules fired:1

Without optimization

Name the hill station with high altitude around Madurai

No of rules fired:2

From the above samples it is clear that the rules fired for the optimized query are less than the rules fired for the unoptimised one.

From the analysis done, it can be inferred that the semantically equivalent queries that are formed by attaching fragments of integrity constraints can be more efficiently processed than the original one. Thus the optimization done on the query saves time and search space. The Performance evaluation is shown in Figure 7.2 and table 7.1.
Table 7.1 Semantic Query Optimization – Tourist Domain

<table>
<thead>
<tr>
<th>Query Number</th>
<th>Time Taken (Unoptimised) In milliseconds</th>
<th>Time Taken (Optimised) In milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>310</td>
<td>230</td>
</tr>
<tr>
<td>Q2</td>
<td>220</td>
<td>60</td>
</tr>
<tr>
<td>Q3</td>
<td>170</td>
<td>110</td>
</tr>
<tr>
<td>Q4</td>
<td>360</td>
<td>290</td>
</tr>
<tr>
<td>Q5</td>
<td>240</td>
<td>180</td>
</tr>
</tbody>
</table>

Figure 7.2 Semantic Query Optimization – Tourist Domain

The performance analysis of the query optimization has also been studied with employee and student databases. Graphs are plotted with the number of rules fired for each query before and after optimization as shown in Figures 7.3 and 7.4 and Tables 7.2 and 7.3 for student and employee databases. The time for the execution of the query is noted down before and after optimization.
List of queries for Employee database

1. Retrieve the Department number, Manager ID, Floor number and Volume of the item motor
2. Retrieve the Manager's ID and his salary at the age of 55
3. Retrieve the superior of an employee who is a supervisor of age greater than 40
4. Retrieve the employee who is a worker of age greater than 40.
5. Retrieve the item and rate whose volume is less than 20000.

Table 7.2 Semantic Query Optimization – Employee Database

<table>
<thead>
<tr>
<th>Query Number</th>
<th>Time Taken (Unoptimised) in milliseconds</th>
<th>Time Taken (Optimised) in milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>170</td>
<td>60</td>
</tr>
<tr>
<td>Q2</td>
<td>210</td>
<td>0</td>
</tr>
<tr>
<td>Q3</td>
<td>330</td>
<td>100</td>
</tr>
<tr>
<td>Q4</td>
<td>220</td>
<td>60</td>
</tr>
<tr>
<td>Q5</td>
<td>360</td>
<td>220</td>
</tr>
</tbody>
</table>

Figure 7.3 Semantic Query Optimization – Employee Database
List of Queries for Student Database

1. Retrieve the section and the age of the staff vinotha.
2. Retrieve the name, ID, Section ID of students whose score is 20.
3. Retrieve the faculty ID and section ID of all faculty members whose age is 20.
4. Retrieve the student ID enrolled in a particular subject.
5. Retrieve the faculty ID, Subject and minimum marks allotted to students.

Table 7.3 Semantic Query Optimization – Student Database

<table>
<thead>
<tr>
<th>Query Number</th>
<th>Time Taken (Unoptimised) in milliseconds</th>
<th>Time Taken (Optimised) in milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>390</td>
<td>150</td>
</tr>
<tr>
<td>Q2</td>
<td>300</td>
<td>120</td>
</tr>
<tr>
<td>Q3</td>
<td>450</td>
<td>190</td>
</tr>
<tr>
<td>Q4</td>
<td>280</td>
<td>60</td>
</tr>
<tr>
<td>Q5</td>
<td>360</td>
<td>260</td>
</tr>
</tbody>
</table>

Figure 7.4 Semantic Query Optimization – Student Database
The optimization performed on the three domains is analyzed based on the time taken for the query execution (Manjula et al. 2002b). The improved performance is due to the result of reduction in search space due to the addition of constraints to the database. Therefore, it can be readily seen that the semantically equivalent query can be processed efficiently by reducing the search space, which in turn reduces the time of query execution.

The usual syntax of predicate logic does not support a structured representation of knowledge. Moreover, predicate logic does not allow for an adequate treatment of incomplete and contradictory knowledge or of subjective and time dependent knowledge. One aspect of a structured representation is that semantically related information should also syntactically be grouped together. This structured representation allows faster retrieval. In order to overcome the structured representation problem in predicate logic (Baader 1999), another approach where the knowledge of the document has been represented using the description logic (DL) formalism has been described. The concepts are represented as a DL hierarchy where the representation is structured. The greatest advantage of representing knowledge in DL is that the system is now able to handle inconsistency and incompleteness present in the user queries. This is useful to the naive user who is unfamiliar in framing structured queries.

7.5 DESCRIPTION LOGIC (DL) REPRESENTATION

Keyword-based search results in poor recall due to the fact that it ignores specialization/generalization and synonym handling. This search also exhibits low precision, that is, retrieval with a large amount of irrelevant
information. This is due to the fact that the structure of the knowledge representation of the document is not well established. The work described in this thesis attempts to capture effectively the knowledge of the document using knowledge representation formalism Description Logic (DL) and represents the concepts as a DL hierarchy. The knowledge of the document is separated into two structures, the Terminological box, TBox containing the concepts and assertion box, ABox containing their instances. This methodology of the design of the DL system enables an effective means for both Information Extraction and Retrieval.

In this work the frequently occurring words of documents are mapped onto an appropriate domain hierarchy using the DL formalism. Using the algorithm described in this thesis, the DL representation of the template is constructed. The content words of the glosses are taken for the correct sense of the template or indexed words. Using these content words of the gloss, DL representation is formed by connecting the nouns of the content words using AND operators (Manjula et al. 2003c, Aghila et al. 2003d). The reasoning services of DL are then used to provide enhanced retrieval.

7.5.1 Fundamentals of DL

The characteristic feature of Description Logic resides in the constructors for establishing relationships between concepts. A concept is interpreted as a set of individuals and roles are interpreted as sets of pairs of individuals. It is worth mentioning that intersection, union, and complement of concepts have also been referred to as concept conjunction, concept disjunction and concept negation, respectively, to emphasize the relationship to logic. The
Description logic hierarchy is defined for the domain chosen. DL provides reasoning services such as consistency, instance checking, quick updating, and subsumption.

DL represents information in terms of descriptions, and basically consists of the Terminological box, TBox containing concepts and assertion box, ABox containing their instances or assertions. Concept is a named description. Concepts are fully defined in the knowledge base by the description associated with them. Intuitively, concept denotes a collection of individuals. Roles are ordinary binary relations that relate individuals to each other. Rules consist of an antecedent and a consequent, which are both descriptions. When the antecedent concept applies to the state of an individual, the rule is "fired" and the consequent concept may also be asserted to apply to the individual. Individuals are specific instances of concepts. Each individual has a name and variable states.

One key element of a DL knowledge base is given by the operations used to build the terminology. Such operations are directly related to the forms and the meaning of the declarations allowed in the TBox. The basic form of declaration in a TBox is a concept definition, that is, the definition of a new concept in terms of other previously defined concepts.

For example, a cricketer can be defined as a person-playing cricket by writing this declaration:

\[ \text{Cricketer} = \text{Person AND cricket} \]
The ABox contains extensional knowledge about the domain of interest, that is, assertions about individuals, usually called *membership assertions*.

For example,

Cricket AND Person (Sachin)
states that the individual Sachin is a cricketer.

Complex concepts are built up from identifiers using *description constructors*. The following are the basic *description constructors*:

The **and** constructor

This constructor forms the conjunction of some number of descriptions. For example, a Vegetarian Person may be defined as:

\[(\text{and Vegetarian, Person})\]

This means that Vegetarian Person is someone who is both a Vegetarian and a Person.

The **all** constructor

An **all** constructor, also called as *value restriction*, specifies that all the fillers of a particular role must be individuals described by a particular description.
For example, the instances of \textbf{(all food, meat)} must have all their fillers for \textit{food} be instances of \textit{meat}, for example, \textit{beef}.

7.5.2 DL for conceptual hierarchy

For each word in the bag of words, the gloss information of that word is extracted. The content words of the gloss along with its Parts-of-speech are then identified. In this work the description of the concept is obtained by connecting the nouns using the AND operator. If 'to' or 'some' precedes any verb then connection is made by establishing the SOME operator followed by the verb name and followed by the noun. For example the concept sports-person can be automatically defined if WordNet gives the following words

\[
\text{(woman(n), or, man(n),some, engages(v), sports(n))}
\]

Define-concept sports-person(

AND (man OR woman)

(SOME engages sport)

DL Representation for the content word, temple, in the tourist domain, the gloss taken from the Wordnet is as follows:

An edifice for the worship of a deity

The output of the gloss is

Edifice/n worship/n deity/n

The DL representation is as follows:

Define-concept temple

AND (edifice)

(SOME worship .deity))
The DL representation for indexed words is done automatically using the following algorithm.

1. Get the content word along with its gloss
2. Check if the content word is already defined as concept/primitive concept/atomic concept
3. If it is an atomic concept or primitive concept, go to step 5.
4. Else define it as a primitive concept in the following way by maintaining the order of words from the gloss of the wordnet:
   i) Make the conjunction of words having noun tag
   ii) If verb or 'to' comes use SOME/ALL operator followed by verb-name, followed by the noun word.
5. Make instantiation in the Abox of that document.

Finally the bags of words extended by lexical and conceptual knowledge are now a bag of set of words. Each set contains the synset of that word and its conceptual words.

**Conceptual taxonomy for the sports domain**

Define-concept sports Athletics(competition AND physical exertion AND active-diversion)
Define-concept sports-person(AND(woman OR man))
   (some engages sport)
Define-concept tournament(AND (sports AND competition AND contestants))
   (some play games)
   (some decide winner)
Define-concept competition (AND (occasion AND WIN))
(some selected contestants))

Define-concept win(AND (contest OR competition)
(victorious))

Define-concept coach(AND (incharge AND training)
(athlete OR team))

Define-concept medal(AND (award win championship))

Define-concept captain(AND (sports team))

Define-concept trophy(AND (token victory))

Define-concept championship(AND (some win first place)
(competition))

Once the concepts are represented in Tbox, the values are filled in A box by the heuristic rules discussed in the previous chapter. The value instantiation process is explained in the next section with a set of documents in the domain under consideration.

7.5.3 Instances Extraction

The document whose information is to be extracted is fed as input in to the preprocessing phase. The preprocessing phase removes the stop words and the most frequent words are obtained. These content words form the concepts, which define the document knowledge. The content words are used for extracting specific entities from the document. The user defines a bag of words, which forms the entity to be extracted from the document. With these bags of words, their existence is checked in the predefined domain hierarchy. If they exist, the documents corresponding to them are taken for the extraction of
the specified pattern. These documents are preprocessed and fired with heuristic rules for determining their relation with the specified concept words. Thus the extraction of specific pattern is done. If the user needs information about “India one day world cup match in Harare”, then the specific content words for extracting information are

“India, World cup, Harare”

Let us consider the sample example document

“World cup is major sporting event in the cricket. World cup is conducted four years once. All the cricket playing countries participate in the match. Sachin has world cup record of highest run. Wasim Akram is the highest wicket taker in the world cup. India plays with South Africa in World cup semifinals on Wednesday, online booking for this match can be done at the following site www.cricinfo.com”.

The document is analyzed for the occurrence of the words India, Harare, and World cup. Since India is an instance of the concept Country, analyze begin with the ABox of the document if the specified information in the ABox is not found then TBox is analyzed for its presence. This method of splitting into ABox and TBox is of great use when the extraction is to be done for a large document by way of saving the time and making the analyze more effective. The ABox of the presented document is

Sachin (AND (India AND Batsmen (ALL consider Finest Batsmen))
Wakim Akram (AND (World cup AND highest wicket))
India (AND (South Africa AND world cup)(ALL play) (book AND match)(ALL do)(AND site))

By analyzing the ABox, the last sentence contains all the three content words, hence the sentence is extracted as

*India plays with South Africa in World cup semifinals on Wednesday, online booking for this match can be done at the following site “www.cricinfo.com”.*

The following heuristic rules are fired

If V Rt NP and P is “in” then Relation Rt is place and so the place is found to be South Africa, Also

NP + P + NP + V ‘day’ if P is ‘on’ which informs that the day is Wednesday.

Hence the instance extracted for the specified words are South Africa and Wednesday. This informs the user that India plays with South Africa on Wednesday. Thus the user is provided with the relevant information.

7.5.4 Use of DL for information retrieval

The DL system attempts to represent the user query in DL and then does the preprocessing part. The use of DL as high-level data description languages has several advantages, including the use of reasoning support for
DL to assist in query formulation at the user-interface level and in query translation and optimization. DL query language is concerned about both incompleteness and inconsistency of query plans.

In the DL query processing system, the query is represented as expressive DL and then analyzed for retrieval. Even if the user gives inconsistent query the DL representation enables the system to retrieve the documents. Inconsistency is one, which is irrelevant to the domain, or which is not present but the user unknowingly tries to get the result. While in the previous systems if the document is not retrieved, the user searches for the same with other sets of words; but the proposed system retrieves “no such data of that kind”. This enables the user to know that no such data exists in the particular domain. The system is enabled to handle incomplete query, which is given by the user with insufficient information for search, and the query is translated into DL and handled. Other systems ask the user to try out with a different set of key words but the proposed system handles it, by retrieving a document of more general concepts.

The first step in this process is the transformation of the user query into Description logic. Again this requires preprocessing the query. The content words in the query are tagged with Parts of Speech. The nouns in the query are connected with the DL constructor “AND”. The verb is connected either with “SOME” or “ALL”. The nouns are used for generating the hierarchy. The hierarchy of the query is checked with defined hierarchy, since the hierarchy generated as random placement of nodes and each node is shuffled according to the predefined hierarchy. Thus the matching document is retrieved.
Consider the following user query

“Give the details of Cricketer Sachin”.

The concept words are Cricket, Sachin, and Detail.

The DL generated as  
Detail AND (Sachin AND Cricket) (SOME give). The hierarchy generated, is compared with the existing hierarchy, which has ‘Cricket’ as parent node and ‘Country’ as child node, which in turn has the ‘Name’ as child node. Further ‘Name’ as ‘Team member’ where the individual Sachin lies. Thus by traversing the nodes the document relating to a specified query is retrieved.

The individual Sachin is an instance of the concept ‘Team member’ and hence the document is retrieved from the ABox. The entire query handling first checks the ABox, since it contains more specific information about the individual. If the query is general and its instance does not exist in the ABox like ‘Indian cricket team’, then checking begins in TBox. Even if the query contains only concepts, the concepts are translated into DL and then they are processed in a similar manner.

7.5.5 Inconsistent Query Handling

The uniform representation of the domain knowledge using DL allows handling erroneous, incomplete and inconsistent queries.

Consider the user query,
List the goals scored by Sachin
This is interpreted as Goals AND Sachin

The Individual Sachin is a instance of the concept of ‘Team Member’ as per the usual procedure; first the search is carried out in the ABox which indicates the presence of the individual Sachin but the career record of Sachin is related to runs and not to Goals, and Goals are associated with football and Sachin is not related to football and hence the user query contradicts. The output produces “no such data”. Thus the system handles inconsistency in the user queries.

7.5.6 Incomplete Query Handling

Incomplete query is a query with insufficient information for handling it. The system handles this incompleteness also. For the given query,

“Give details of Maradona’s winning goal”
This query has been transformed into
(Maradona AND goal)

Where Maradona is an ABox entity and goal is a concept in TBox. The built concept hierarchy indicates that the goal is associated with some of the games like Hockey, Football etc. The other systems retrieve all the documents relating to ‘goal’ including the documents of hockey which are utterly irrelevant. For all those games the system will automatically check their associated Abox assertions to find out the instance name ‘Maradona’. Only the Abox of Football has the instance ‘Maradona’ and hence the document relating to ABox instance Maradona is retrieved which is the one the user needs to retrieve.
Also if the user gives a query in the form of concept words (nouns) and not in the form of a sentence, even then the system retrieves the document as in any other system. The system retrieves all the TBox documents related to the user words as TBox has more general information. Thus the system handles queries in a more effective way.

7.5.7 Performance Evaluation

By the use of precision and recall, the effectiveness of the system in retrieving the document and relevance of the retrieved documents are evaluated. The performance is compared with a traditional keyword matching document retrieval method.

The documents are to be classified based on the content words of the user query. For each word the threshold is fixed. The documents are categorized as relevant which is above the threshold value. The proposed system is compared with the ordinary content word based retrieval where the documents are retrieved based on the content words. The documents retrieved are again categorized into relevant and not irrelevant by the constraint described above. For example, the query is “give cricketer Sachin career details” the proposed system retrieves document relating to ‘Sachin’, which may have his personal details and other career details. The document is considered as relevant if all the content words are above the specified threshold. The content word based retrieval retrieves cricket related documents, Sachin related documents, which are not all relevant. Thus the performance of the proposed DL system is higher compared to the word based system.
Sample Query 2:

DL system

<table>
<thead>
<tr>
<th>Query</th>
<th>give the cricketer details</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of document retrieved (A)</td>
<td>7</td>
</tr>
<tr>
<td>No of relevant documents retrieved (B)</td>
<td>6</td>
</tr>
<tr>
<td>No of relevant documents not retrieved (C)</td>
<td>1</td>
</tr>
<tr>
<td>Precision (B/A)</td>
<td>0.857</td>
</tr>
<tr>
<td>Recall (B/(B+C))</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Word Based System

<table>
<thead>
<tr>
<th>Query</th>
<th>give the cricketer Sachin details</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of Documents retrieved</td>
<td>28</td>
</tr>
<tr>
<td>No of relevant Doc retrieved</td>
<td>5</td>
</tr>
<tr>
<td>No of relevant Doc</td>
<td>7</td>
</tr>
<tr>
<td>Precision</td>
<td>0.178</td>
</tr>
<tr>
<td>Recall</td>
<td>0.71</td>
</tr>
</tbody>
</table>

Similar queries are given to the DL system and the word-based retrieval. The precision and recall values of both the systems are calculated. The results for a sample of 5 queries are tabulated in Table 7.4.
Table 7.4 Results of DL based Systems

<table>
<thead>
<tr>
<th>Query</th>
<th>Word Based</th>
<th>DL Based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>1.</td>
<td>0.212</td>
<td>0.68</td>
</tr>
<tr>
<td>2.</td>
<td>0.178</td>
<td>0.85</td>
</tr>
<tr>
<td>3.</td>
<td>0.27</td>
<td>0.90</td>
</tr>
<tr>
<td>4.</td>
<td>0.19</td>
<td>0.8</td>
</tr>
<tr>
<td>5.</td>
<td>0.2</td>
<td>0.833</td>
</tr>
</tbody>
</table>

The overall precision and recall of the DL system is found to be better than that of word based retrieval. However the recall and precision depends on effectiveness of the constructed domain hierarchy and appropriate updating the hierarchy.