An Ontology Based Information Retrieval System

Karthik Pai B. H, Balaji N.

***Abstract**: Ontology provide a structured way of describing knowledge. Ontology's are usually repositories of concepts and relations between them, so using them in information retrieval seems to be a reasonable goal. The main objective in this report is to provide efficient means to move from keyword-based to concept-based information retrieval utilizing ontology's for conceptual definitions [1]. In this paper, we present the skeleton of such an IR system which works on a collection of domain specific documents and exploits the use of a domain specific ontology to improve the overall number of relevant documents retrieved. In this system, a user enters a query from which the meaningful concepts are extracted; using these concepts and domain ontology, query expansion is performed. We propose a system that matches the query terms in the ontology/schema graph and exploits the surrounding knowledge to derive an enhanced query. The enhanced query is given to the underlying basic keyword search system LUCENE [2]. In this approach we try to make use of more ontological Knowledge than IS-A and HAS-A relationships and synonyms for information retrieval.*

***Keywords**: Ontology, Semantic, DSA, IR System.*

I. INTRODUCTION

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored as a document corpus). In the olden days, information retrieval used to be an activity that only few people were engaged like reference librarians, but over the years, the volume of information available has increased tremendously. Unfortunately, the unstructured nature and huge volume of information has made it difficult for users to sift through and find relevant information. Therefore, the role of searching applications has become very crucial. Numerous information retrieval techniques have been proposed to help deal with this problem. Information retrieval systems based on these commonly used keyword-based techniques are many a times difficult for ordinary or naive users as naive users often have difficulty in expressing their information needs to get back relevant results [3]. This problem arises because a precise representation of user's information need in the terms that system uses exactly is not possible to achieve. One more reason for this problem is that search terms applied by the user may be different form the keywords used by the system.
Therefore, our main objective is to overcome the drawbacks of a traditional keyword based or full text search engine that do not consider the underlying meaning or user's intent in information retrieval systems. One such open source full text search engine for information retrieval is LUCENE. We move form keyword based search to semantic search or conceptual search with the help of underlying domain specific ontology. We present a detailed account of our approach in the subsequent sections. In this section, the basics and overview of any Information Retrieval System is presented.

A. Main Components of an IR System

Any Information Retrieval system is supported by the Retrieval process which involves three basic processes, which are as follows:
- The representation of the content of the documents,
- The representation of the user's information need, and
- The comparison of the two representations.

The processes are visualized as follows.

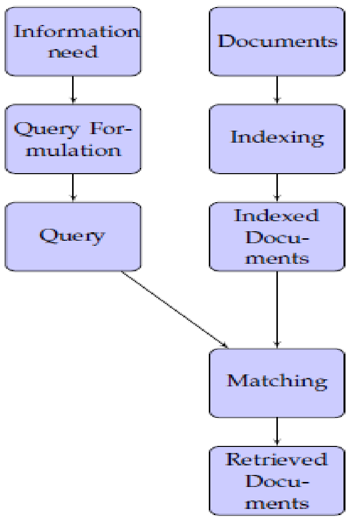

Figure – 1: An Information Retrieval System

Representing the documents is usually called the indexing process. The process of representing the information need of a user is often referred to as the query formulation process.

Manuscript received on May 25, 2020.
Revised Manuscript received on June 29, 2020.
Manuscript published on July 30, 2020.

* Correspondence Author
Dr. Karthik Pai B H*, Department of Information Science and Engineering, NMAM Institute of Technology, Nitte, Karkala, India. E-mail: Karthikpai@gmail.com
Balaji N, department, Department of Information Science and Engineering, NMAM Institute of Technology, Nitte, Karkala, India. E-mail: balaji.hiriyur@gmail.com

The resulting representation is the query. The comparison of the query against the document representations is called the matching process. Retrieval strategy refers to information retrieval model.
Retrieval strategies assign a measure of similarity between a query and a document. Any Information Retrieval system is based on Information Retrieval process.

II. LITERATURE SURVEY

In this section we present various information retrieval strategies that have been studied for the purpose of this project. We emphasize more on those models on which LUCENE is built.

A. *Information Retrieval Models*

Every IR model transforms documents into an appropriate representation which are used for retrieval of relevant documents. The most important and state-of-the art models for information retrieval are the Boolean model, the Statistical model and the Linguistic and Knowledge-based models.

B. *Boolean Model*

The Boolean model is one the first models of information retrieval model. It is a simple retrieval model based on set theory and Boolean algebra. In this model, the queries are specified as Boolean expressions which have precise semantics. For example, the query term "finance" defines the set of all documents that are indexed with the term *finance*. In this model, the operators of George Boole's mathematical logic - logical product AND, logical sum OR and logical difference NOT - can be combined along with query terms and sets of documents to form new document sets.

C. *Extended Boolean Model*

Retrieval using Boolean model is simple and elegant. However Boolean model has no provision for term weighting. So, no ranking of the answer set is possible. As a result the size of the output set may be too large or too small which is not desirable. Because of this problem, modern information retrieval system are no longer based on Boolean model. Smart Boolean approach and extended Boolean models (for example: **P-norm** and **Fuzzy Logic** approaches) provide relevance ranking to users.

D. *Vector Space Models*

Vector space model requires that retrieval objects are modelled as elements in a vector space. In this model, terms, documents, queries, concepts are all represented as vectors in the vector space. Unlike the Boolean model, here we don't use binary weights but assign non binary weights to index terms in queries and documents. These terms are used to compute degree of similarity between document and query. Here the similarity measure used the cosine of the angle that separates the two vectors x and y, where x represents the documents index representation and y represents the query.

E. *Probabilistic Models*

Classic probabilistic models also known as binary independence retrieval model, was introduced by *Roberston* and *Spark Jones*. The probabilistic model attempts to capture the IR problem within a probabilistic framework. It tries to estimate the probability that a user finds a document d_j relevant [4]. It assumes that the probability of relevance depends on the query and document representations only.

III. TOOLS REQUIRED

In this section we discuss about Data Structures and Algorithms Ontology. We limit our discussion to domain specific documents in this paper.

A. *Data Structures and Algorithms Ontology*

We use Data Structures and Algorithms Ontology by [5]. It describes various data structures, their properties such as time complexity, space complexity. It has a total of 88 concepts, 24 object properties, 12 data properties with an axiom count of 665 out of which 405 are logical axioms and 249 are declaration axioms.

B. *WordNet*

WordNet is a huge lexical database for English. It was originally created at Princeton University [6]. Words are the units in WordNet, as the name indicates, though it contains idiomatic phrases, compounds and phrasal verbs. The main purpose of WordNet was to make huge amounts of lexical knowledge available and also a well-defined structure that could support linguistic research better than traditional dictionaries. The foremost idea behind WordNet was to organize lexical information in terms of meanings, rather than word form. This facilitates in moving from traditional dictionary to a lexical resource which incorporates semantic relations between words. In WordNet nouns, adjectives, adverbs are grouped into sets of synonyms called synsets. In our paper we make use of WordNet to fetch synonyms of query terms in order to match these concepts in ontology.

IV. PROPOSED SYSTEM ARCHITECTURE

In this section, we present a detailed account of each and every module used in this system.

A. *Text Extraction Module*

We convert Information from various sources such as HTML pages, pdf files etc., into textual format. This step is essential because LUCENE indexes and searches only information that is in textual format only. This constitutes our document corpus. For the purpose of this project we have created a document corpus which comprises information regarding various data structures, their properties such as space complexity, time complexity. Our document corpus includes information about Array, 2D Arrays, LinkedLists, ArrayLists, Graphs, Trees, and Heaps etc. Information about each of the data structures have been gathered from various sources such as Wikipedia, standard textbooks etc.

B. *Analysis Module*

Indexing text directly is computationally huge task so no search application indexes text directly. In this step every document is analyzed prior to indexing. Analysis is the process of breaking down text into individual atomic elements called tokens. A token can be viewed roughly as a word in the language. The analysis step determines how the textual fields in the documents are divided into tokens.

Here there are many challenges to be handled such as how are compound words handled? Should synonyms also be considered for original token stream so that a search for "laptop" should also return documents mentioning notebook. Though LUCENE provides a rich and wide variety of built in analyzers, but often one size does not fit all requirements. One of the most crucial steps in building any search application is to choosing the right analyzer that satisfies all requirements. For this purpose we have developed our own custom analyzer that performs the following tasks.

- Removal of white spaces,
- Perform Stemming,
- Perform Parts of Speech Tagging,
- Extract Nouns as descriptors of concepts which are to searched in Ontology, and
- Extract Prepositions and remove all stop words other than prepositions which are not present in ontology.

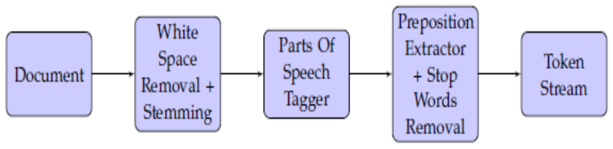
Document
White Space Removal + Stemming
Parts Of Speech Tagger
Preposition Extractor + Stop Words Removal
Token Stream

Figure – 2: Steps in Analysis Phase

C. *Indexing Module*

After the analysis process, LUCENE stores the analyzed input in data structure called inverted index that makes efficient use of disk space and allows faster keyword look-ups. This data structure is called inverted index because it uses tokens extracted from input documents as keys for look up instead of considering documents as keys for look up.
From a high level perspective a collection of segments form a LUCENE index. A segment is nothing but a standalone index which holds a collection of indexed documents. A new segment is created whenever the writer flushes buffered added documents and pending deletions into the directory. At search time, each segment is visited separately and the results are combined. Also the contents and path of each file document are stored in the index file. For each token, meta-data such as position of token within the document, how many documents contain a token etc., are extracted and stored in the index file.

V. QUERY ANALYSIS

User enters query in natural language through search interface. The query is passed through Analysis Module. This module performs following tasks.

A. *Spell Check*

The query which the user inputs is first checked for spelling mistakes as our document corpus does not contain such miss spelt words. A Spell checker scans and extracts the words contained in the text. Next it compares each word with a known list of correctly spelled words from a dictionary.

B. *White Space Removal*

White spaces are removed from the query as they do not contribute to distinguish any document.

C. *Stemming*

Many a times, user specifies a word in a query but only a variant of this word is present in relevant document. Plurals, gerund forms, past tense suffixes are examples of syntactic variations of the same word. This will prevent a perfect match between a query word and a document word. So to overcome this problem, we substitute the words by their respective stems or head words. A stem is a portion of the word which is left after removal of affixes (suffixes and affixes).An example of a stem is the word connect which is the stem for connected, connecting, connection and connections. The process of stemming improves performance of information retrieval as it reduces variants of same root word to a common concept. Another important benefit we get by stemming is that it reduces the size of the index structure as the number of distinct index terms is reduced.

D. *Parts Of Speech Tagging*

Each word in the user entered query is passed through a parts of speech tagger. At the end of this process, a tag with its syntactic nature (adjective, noun, pronoun, conjunction). We extract nouns from query words and form a Descriptor List. We choose nouns for forming the descriptor list as nouns carry more semantic information than other parts of speech and more importantly the concept names in the ontology are predominantly nouns.

E. *Remove stop words other than prepositions that are present in Ontology*

Words which occur very frequently among the documents in the corpus are not good discriminators. In fact a word which occurs in 80% of the documents is useless for retrieval purposes. So, we eliminate stop words with the objective of filtering out words with very low discrimination values. Such words are referred to as stop words. Words such as is, are, the, of, etc., are some common examples of stop words. Elimination of stop words is very beneficial as it reduces the size of the indexing structure reasonably. In our approach we remove we remove all stop words but retain those prepositions that are present in the domain Ontology. Relationships in ontology's mostly contains prepositions such as teaches in, is son of, etc. Removal of them may prevent an exact match, so we retain prepositions that are present in ontology in an attempt to improve efficiency of retrieval system.

VI. KEY WORD QUERY AND QUERY EXPANSION USING ONTOLOGY

In this stage the query is expanded semantically using knowledge from the domain Ontology. The main objective here is to discover new relationships between query terms from the ontology. Query Expansion is the process of aggregating query terms with additional terms. Each concept in the descriptor list which is obtained in the Query Analysis phase is compared with concepts in ontology and those concepts for which there is a match are separated into another list C. The figure – 3 describes query.

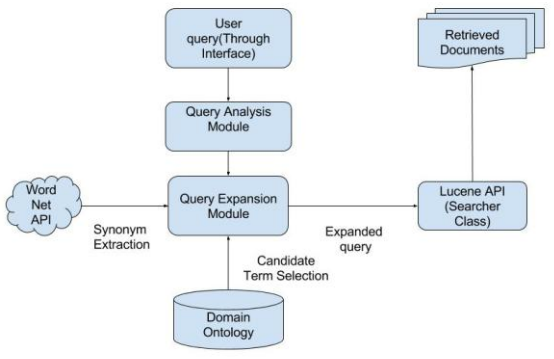

Figure – 3: Query Expansion Module

Let K be the set of Keywords $\{k_1, k_2, \dots, k_m\}$ that matched with some concepts in Ontology which is denoted as $C = \{c_1, c_2, \dots, c_j\}$. Query Expansion reformulates the given query by appending to it a set of keywords $k_{m+1}, \dots, k_{m+n}$ that are obtained from the set of Concepts C in Ontology. Basically in this phase query terms are replaced by collection of some additional terms and original terms. This enhanced query terms are used to generate more relevant results.

A. *Candidate Term Selection from Ontology*

In this section, we define how a query term is expanded when the query term itself or its synonym which is fetched from WordNet is matched to a concept in ontology. Here we consider distance from the original matched concept as an important metric in determining which concept to select for aggregation. The proposed algorithm traverses all IS-A and HAS-A relationships by which we get sub concepts and super concepts respectively.

In Algorithm, Wu & Palmer similarity measure [6] is used to calculate similarity by considering the depths of the two concepts (s1 and s2), along with the depth of the Least Common Subsumer (LCS). The formula score is given as follows.

$$Wu\_Palmer_score(c\_i,c\_j) = \frac{2 \times LCS(c\_i,c\_j)}{(length\_min(c\_i,c\_j) + 2 \times LCS(c\_i,c\_j)}$$

It can be noted from the above equation that 0 <=score <= 1. As the depth of the least common subsumer is never zero the score is always < 0. As can be easily seen, if the input concepts are the same, the wupalmer score is 1.

The Algorithm for candidate term selection is given below,

Input: $C = \{c_1, c_2, \dots, c_j\}$ the set of ontological concepts matched with query terms or synonyms of query terms. Term_Weight (tw), $tw(c_i) = 1$.

Output: Q = $\{k_1, k_2, \dots, k_m\}$

Algorithm 1 Concept Selection

```
1:  depth ← 1
2:  Q′ ← {}
3:  for c ∈ C do
4:      for depth =1 : 4  do
5:          C′=getConcepts(c_i,depth)
6:          for c″ ∈ C′ do
7:              sim(c″)=Wu_Palmer_Sim(c′, c″)
8:              tw(c″)=sim(c″)
9:          end for
10:         if ( thentw(c″ ≥ Threshold))
11:             Add c″ to Q′
12:         end if
13:     end for
14: end for
```

VII. ONTOLOGY BASED QUERY EXPANSION ALGORITHM

Whenever a keyword query gets mapped to a concept in Ontology, it is expanded as follows. When the keyword is not present in Ontology, its synonyms are fetched from WordNet and is checked whether it is present in ontology.

A. *Case 1: (Concept)*

Whenever the query term gets mapped to a concept in the ontology, the query is expanded in terms of its properties, sub classes and instances that are inferred from ontology. This way of expansion guarantees semantically relevant documents are retrieved even if the query term is not present in the document. As we are expanding the query term in terms of the knowledge inferred from ontology also ensures that the documents are retrieved in the right context. For example a query on graph term should also retrieve the documents which speak of data structure with vertices and edges even in the absence of the query term graph. In the above example the expanded query now retrieves the documents which discusses the properties of graph like edges, vertices, etc., or sub-classes of graphs like Directed graph, un-directed graph, or instances of graph like Simple graph, Multi-graph, Null graph. If the query term gets mapped concept named Graph in Ontology, it is expanded as follows.

QUERY Term: Graph

Expanded Query:

{(ADT and Edge and Graph_node and Graph_Operation) or
(Graph and (Applications or Graph_Operations)) or
(Acyclic Graph or
Cyclic Graph or
Directed Graph or
Un-directed Graph) or
(Simple Graph or
Null Graph or Multi-Graph)}

B. *Case 2: (Property)*

If the query term gets mapped to a property name in the Ontology, we expand the query in terms of its domain and range. For a query edge, the expanded query will be in the form of (Graph .and. edge) .or. (Tree .and. edge) or. (edge. and. directed) .or. (edge. and. un-directed) which ensures that documents relevant to edge are retrieved in the right context.

Query Term: Edge
Expanded Query:
{(Graph and Edge) or
(Tree and Edge) or
(Head and Edge) or
(Edge and Directed) or
(Edge and Un-directed)}

C. Case 3: (Instance)

In the case where a query term gets mapped to a concept in the ontology we include it directly in the aggregated list. Whenever the query maps to two terms in the ontology, relationship between them is identified and exploited accordingly.

D. Case 4: (Concept, Concept)

Sibling-of, IS-A, and disjoint-with are the possible relationships that can hold between two concepts. In this case where the two query terms get mapped to a concept and concept, it is expanded as follows. If the two given concepts have a common ancestor, we exploit the specific properties possessed by the common ancestor and expand the query in terms of properties possessed by ancestor along with that of their individual properties. For example in the tourism domain the query Conference Room, Guest Room map to the concepts Conference Room, Guest Room with the parent concept Room. We retrieve documents that discuss these concepts or their parent concept in the context of the common properties such as Name, Internet-Access, Telephone, TV or the specific properties of either of the query concepts. The specific properties of the concept Guest Room are Minibar, Terrace, Balcony, Bed... whereas Projector, Stage, Video Conference system, Screen, are the specific properties of the concept Conference Room. We leave out the documents that do not contain none of the common or specific property terms. When the given concepts do not have a common ancestor, the query is expanded in terms of the intermediate concepts and the connecting properties.
Query Terms: Conference Room, Guest Room
Expanded Query:
{(Room and (TV or Internet Access or Phone or Cleaning Service)) or
(Conference Room and (Speakers or Projector or Stage)) or
(Guest Room and (Terrace or Balcony or Bed))}

E. Case 5: (Concept, Property)

A specific case of Property case where either the domain or range is provided by the user. If the given concept is a domain (range), the expanded query includes the range (domain) of the property. The domain and range are further expanded in terms of their inferred sub-classes and instances. For example Q (Tree- Applications) bring out the results containing the sets Tree .and. Applications .and. Router Algorithms and Tree .and. Applications .and. Traversals. Then the query pull out those containing the instances of the Queue along with the property, Binary Tree .and. Applications, N-Ary Tree .and. Applications.
Query Terms: Tree, Applications
Expanded Query: {(Tree and Applications and Router Algorithms) or
(Tree and Applications and Traversal) or
(Binary Tree and Applications) or
(N-Ary Tree and Applications)}

F. Case 6: (Property, Property)

Given two property terms, we proceed with identifying the concept(s) on which these given properties are defined (in the place of domain or range). Whenever the given properties identify a common concept the query is expanded in terms of the common concept along with these properties. The query term Cleaning Service, Video Conference System corresponds to two properties defined on the concept Conference Room in the tourism domain. Once the concept is identified, we discuss these properties in the context of Conference Room. So the enhanced search string is Conference Room .or. (Cleaning Service .and. Video Conference System).
{(Conference Room) or
(Cleaning Service and Video Conference System)}

G. Case 7: (Concept, Instance)

This can be viewed as a specific case of Concept-Concept where we have one of the concepts referring to a specific instance of a class. Just as in the mentioned case, the given instance could be an instance of given class, or could be an instance of a sibling of class C, where the siblings could be either overlapping or disjoint.

H. Case 8: (Instance, Instance)

If each of the keywords is treated as an individual of a concept, it would also be the one specific case of Case 4 where the relationships between the instances are taken into consideration. Accordingly, the common properties, the specific properties (properties of instances in general) and the concept to which they belong (in case they belong to a common concept) are brought into context. It is possible that these two instances are related through a set of (object property, value) pairs, in which case all the connecting (object property, value) pairs are brought into context.

I. Case 9: (Property, Instance)

If the given property holds on the given instance then the query is expanded in terms of the values of the property for that individual. A query for priority-queue Applications, basically looks for the values of the property Applications in the context of priority queue, which in this case is Heap-Construction. If the terms in the query are not related by any of the relationships discussed above, the keyword based search is performed which can be treated as default case.

J. Searching Module

In this module, index file created in the previous module is used for searching the query. We use LUCENE's index searcher class which searches the index files in the inverted index table and renders results.

VIII. CONCLUSION AND FUTURE WORK

We proposed an architecture for information retrieval in a specific domain. Our objective is to move from traditional keyword based search to semantic search by using a domain specific ontology as reference for concept definitions. An algorithm for query expansion has been proposed which aggregates the query terms by selecting concepts in ontology.

For this concept selection, various cases have been identified and in each case query terms are expanded as mentioned in the above sections. This algorithm uses more knowledge than IS-A and HAS-A relationships and ontology and synonyms for query expansion.

REFERENCES

1. A. Macfarlane J. Bhogal and P. Smith, *A Review of Ontology based Query Expansion, Information Process Management, Elsevier, 2007.*
2. Lucene Search: http://lucene.apache.org
3. Rohit Rathore Priyamvada Singh Rashmi Chauhan, Rayan Goudar and Sreenivasa Rao, *Ontology based automatic query expansion for semantic retrieval in sports domain*. ICECCS, pages 422–433, 2012.
4. Norbert Fuhr. *Probabilistic models in information retrieval*, The Computer Journal, 35, 1992.
5. Vinu E V and Rajeev. *Data structures and algorithms ontology*, AIDB, Research Lab, IIT-M, 2015.
6. Wordnet wordnet 2.1 reference manual, http://wordnet.princeton.edu/man/, Cognitive Science Laboratory, Princeton University, 54, 2005.
7. Z. Wu and M. Palmet, *Verb semantics and lexical selection*, Proceedings of the 32nd Annual meeting of the Association of Computational Linguistics, pages 133-138, 1994.

AUTHORS PROFILE

Dr. Karthik Pai B H obtained Ph. D from Visvesvaraya Technological University, Belagavi, and Karnataka. His research area includes Networks, software engineering, Cyber Security. He is a life member of ISTE and published one patent and various research papers in Scopus indexed journals.

Mr. Balaji N has obtained M. Tech in Computer Science and Engineering from IIT Madras, Chennai. His research area includes theoretical computer science, information retrieval systems in domain specific ontology's. He is a member of IEEE, CSI, and ISTE. He published more than five research articles in Scopus indexed journals.