International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184, Volume 5, Number 2 (April 2016), pp.104-109
© MEACSE Publications
http://www.meacse.org/ijcar

A Novel Approach for Constructing Ontology Based Grounded Index and Context Sensitive Query Processor

Sneh Dalal, Jyoti Verma,Komal and Kumar Bhatia

Computer Engineering Department, YMCA University of Science, Faridabad, India

Abstract

Search engine is an IR tool that user employs to find information on the web. It maintainsthe index of downloaded documents that are stored in the localrepository. Whenever user fires a query, search engine searchesthe index in order to find the relevant documents to bepresented to the user. The quality of the matched resultsdepends on the information stored in the index. Keyword based search engines employ inverted index to find the associated documents according to the keywords present in the query. Generally the inverted index isbased solely on the occurrences of keywords in documents. It has the disadvantage of lower precision rate as it completely ignores the context of documents. In order to improve the efficiency of the searchengine, an improved indexing mechanism to index the webdocuments is being proposed that keeps the context relatedinformation in the index structure. In this proposed technique, Ontology Based Grounded Index is built that considers the context of documents and associate each term with an Ontological class by using underlying ontology of that domain. Similarly a Context Based Query Processor is proposed in which the context of the query is analyzed to return the most relevant results to the user and thus increases the precision rate.

Keywords: Search Engine, Indexer, Query Processor, Context Sensitive, Ontology.

Introduction

Search engine index[1] is one of the main components of the search engine that helps in retrieving relevant documents from the huge repository of World Wide Web. Indexing is the process of parsing and storing data for use by the search engine. It isthe search engine index that provides the results for search queries in the form of pages that are stored within the search engine index. Without a search engine index, the search engine would take considerable amount of time and effort each time a search query was initiated, as the search engine would have to scan every web page or piece of data to find the keywords of the query in the documents. Hence index is created in advance to speed up the query processing task. Inverted index [2] is the most common indexing structure employed by the search engines. Structure of inverted index is in the form of term and documents-ids. Term represents the different possible keywords and documents-ids represent listing which contain list of those documents that contain that particular term. Inverted index ignores the context of documents completely. It lists the documents under the same listing which may have different context. Ontology grounded index divides the documents on the basis of their context. Hence it assists in providing more accurate results.
Ontology [3,4,5,6] can be defined as a formal specification of a conceptualization or, in other words, a declarative representation of knowledge relevant to a particular domain. It can also be understood as a shared understanding of some domain of interest. It provides “well-defined meaning” to the information enclosed in the Web. Further it should be machine understandable. Ontologies contain classes, their instances, data property, object property etc. Ontology of a particular domain contains complete information

about that domain. This information is present in the form of triplets i.e. subject, predicate, object. The most important characteristic of ontology for the present research effort is that their role is considered as a structured form of knowledge representation. Ontology in this system has been used to derive the context of documents.

Literature Review

Many indexing and query processing schemes have been proposed over time. In [9], a double indexing mechanism for search engine based on the campus net has been introduced. It consists of crawl machine, index, Chinese automatic segmentation and search machine. A double indexing mechanism has been proposed, which has document index as well as word index. This document index based on the documents clustering is ordered by the position in each document. When searching for the query, the search engine first gets the document id of the word in the word index, and then goes to the position of corresponding word in the document index. This mechanism is time consuming as the index exists at two levels.

Another algorithm is reordering algorithm [10] in which set of documents are partitioned into k ordered clusters on the basis of similarity measure. Then the biggest document is selected as centroid of the first cluster and n/k (where n is total number of documents and k is the number of clusters) most similar documents are assigned to this cluster. Then again the biggest document is selected excluded the first cluster and the same process repeats. This process keeps on repeating until all the k clusters are formed and each cluster is assigned n/k documents. This algorithm is not effective in clustering the most similar documents. It is possible that the biggest document does not have similarity with any of the documents but still it is taken as the representative of the cluster.

In [11] author proposed threshold based clustering algorithm in which the number of clusters are unknown. However, two documents are classified to the same cluster if the similarity between them is below a specified threshold. This threshold is defined by the user before the algorithm starts. If the threshold is small, all the documents will be assigned to different clusters. If the threshold is large, the documents may get assigned to just one cluster. Thus this algorithm is sensitive to specification of threshold.

Proposed work

Current work focuses on the context of the keywords. Ontology Grounded Index (OGI) and Context Sensitive query processor are the main components of the system.

Building OGI is an offline process. Crawler gathers the documents and after that the indexer builds index in the form of Term and Postings. This index is known as inverted index. Inverted index is then refined to create OGI. OGI contains three fields Term, Ontological Class, Postings. This index classifies the documents on the basis of context. OGI is stored in the repository of the system.

On the other hand query processing module works online. User interacts with query processing module through user interface to get the results. When user fires the query, query processor after performing some necessary operation like normalization, Stopword removal lemmatization etc., analyzes context of the query with the help of ontology files.Detailed architecture of the system is shown in Fig. 1. Main components of this system are as follows:

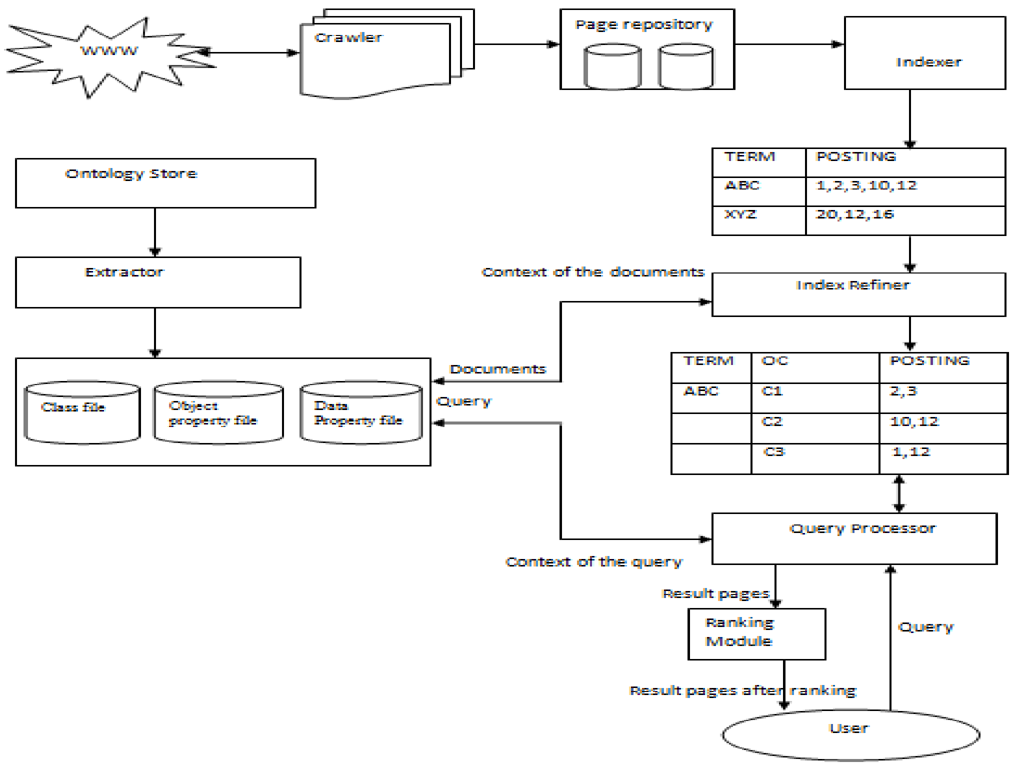

Fig. 1-Architecture of the Proposed System

Ontology store. It contains the Ontology files of various domains with an extension of .owl. Ontology store may be built or packed up with readymade form of concept hierarchy, available for usage. Various tools are also available for building the same where they mine server logs. Here the ontology file is constructed using Protégé 4.3[7].

Extractor. After constructing Ontologies, an ontology extractor is used to extract class file, data property file and object property file from the ontology. These files are used to create OGI.

Class File. Classes are the basic building blocks of ontology. A class is a concept in a domain. There can be different classes present in a single ontology file. Class File describes different classes and their objects.

Data Property File. Data property file consists of data properties of classes. For e.g., Fruit class has the properties taste, variety, shape, origin etc.

Object Property File. Object property file contains information about the type of relationship between two objects.

Indexer. The basic idea behind an indexer module is to construct a data structure with two fields that are Term and Postings.

Index Refiner. Index refiner is a newer component that propels the extension of the index structure. The index refiner module takes the inverted index, ontology store and set of algorithms as input. It then constructs the OGI.

Ontology Grounded Index. Purpose of ontology grounded index is to provide more accurate results by reducing the result look up sphere. It does this, with the help of a new columnar attribute added to the structure named as Ontological Class (OC). OC divides the documents on the basis of their context. Table 1 shows a sample OGI. In this table there are various classes related to a single term like “Apple” term have two classes i.e. “Fruit” and “Manufacturer”. Similarly “Jaguar” is related to two Ontological Classes “Animal” and “Automobile”.

TABLE 1
ONTOLOGY GROUNDED INDEX

Term	OC	Posting
Apple	Fruit	1,4,6,9,35
Apple	Manufacturer	2,7,18,94
Jaguar	Animal	121,59
Jaguar	Automobile	56,78,98
Plant	Business-unit	34,46,213
Plant	Greenery	712,678

Query Processor. Query processor [8] interacts with the user to get the query. Query processors that are not context sensitive, match the query terms in the index file and return corresponding documents.

Context sensitive query processor does not simply match the query in the index file but it also tries to find the context of the query. For example if a user asks about "mileage of jaguar", query processor should be able to make out, by looking in the ontology files, that the user is asking about jaguar car and not about the jaguar animal because mileage is the property of car. By analyzing this context, it can provide more efficient results. But simple query processor does not analyze the context of query and simply returns all those documents that contain keywords jaguar and mileage. It first checks term jaguar in the index and finds its corresponding documents and then checks mileage and finds its corresponding documents and perform the AND operation on both the document lists and returns resultant documents.

Consider another example, in simple query processor if user fires a query such that its keywords are not present in the index, query processor does not return any result in this case. But in proposed system if query processor does not find the query terms in the index, it checks the ontology file to find out if that term is present in the ontology. If it finds that term in the ontology, it finds other related terms and kind of relationship between those terms. Then it returns the documents containing those terms which are related with the fired query terms. For example if a user fires 'IOS' query but 'IOS' is not there in the inverted index then context sensitive query processor will look in the ontology to find out if 'IOS' is related to any other term. If it finds out other terms that are related to 'IOS', it will return the documents containing those terms that are related to 'IOS' because they might be useful to the user.

Algorithms

1.Ontology_Grounded_Index

```
Input: Inverted index, Ontological Class File, Ontology File
Output: OGI (ontology grounded index)
Step 1: Pick up terms from inverted index one by one until all terms are done
Step 2: For each term check whether Ontological classes exist or not by looking in the
ontological class file
  Step 2.1 If ontological classes exist for that term
Get documents and classes related to that term
For each document related to the term do
Correct_Class=DOCUMENT_CHECK (document, Classes)
Make an entry in the OGI for the term under Term field, Correct_Class under
Ontological Class field and document Under Postings field.
else
 Get documents related to the term. Make an entry in the OGI for the term under Term
field, under the Ontological Class field make entry of default class and add all documents
related to the term under Posting field.
endif
```

```
2.Document_Check(document, classes)

Input :document,  different possible classes
Output : Correct Class to which document belongs
Step1:For each class do  From the ontology file check the properties of the class and
increment value of Count for that class each time a property matches to the document
Step 2:Correct_Class = Class for which count value is maximum.
Step 3: return Correct_Class.
```

```
3. Query_processor

Input    :Q where Q is Query Fired by the user.
Output :D_i (i=1 to n)where D_i are Documents that are relevant to the user.
Step1: Tokenize the Q
Step 2:preprocess the query like StopWord Removal, normalization, lemmatization etc.
Step 3:Scan each token to determine to which possible classes the query belongs to.
Step 4:Analyze the query by comparing it with the properties of the possible classes to
determine the correct class of the query
Step 5:Return the documents related to the correct class and query terms from the
Ontology grounded index.
```

Conclusion and Future Work

Keyword based indexing and query processing techniques seems to be inefficient as it completely ignores the context of the documents and the query. It can include those documents in the result pages which include some keywords of query yet their contexts are entirely different from the fired query. Hence in those cases precision of result set will be quite low.

But proposed system uses ontology to determine the exact context of the query. Ontology based grounded index will assist in providing more accurate results and in narrowing the result look up sphere, thus reducing the number of irrelevant results considerably by analyzing the context of the term. It has advantage over other web page classification techniques also. Ontology based query processor will not include those documents in the result set which contain keywords of query but different in context to the query context. Proposed system is cost effective as well as fairly simple because it uses only the ontology to decide the exact context of the documents and the query.

In future this technique can be combined with other techniques like log history analysis, personalization etc. to determine the context of ambiguous queries in less time. In cases where it is not possible to determine the context of the query, user history can be used to determine the context of the fired query. Ontology based ranking can also be developed in future, in which context of the documents will play an important role in calculating the rank of the documents.

References

[1]http://en.wikipedia.org/wiki/Search_engine_indexing.

[2]http://en.wikipedia.org/wiki/Inverted_index.

[3] A Guide to Creating Your First Ontology:Natalya F. Noy and Deborah L. McGuinness; Stanford U. Report.

[4] T. R. Gruber. A translation approach to portable ontologies.Knowledge Acquisition,5(2):199-220, 1993.

[5] Mike Uschold MichaelGruninger AIAI-TR-191, "Ontologies Principles Methods and Applications", February 1996

[6] Nicola Guarino ITSC-CNR, Laboratory for Applied Ontology, Daniel Oberle SAP Research CEC Karlsruhe and Steffen Staab University of Koblenz-Landau ," What Is an Ontology?".

[7] HolgerKnublauchet. Al ," A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools".

[8] Dr. G. R. Bamnote and Prof. S. S. Agrawal, "Introduction to Query Processing and Optimization", International Journal of Advanced Research in Computer Science and Software Engineering, 7 july, 2013.

[9] Changshang Zhou, Wei Ding, Na Yang, "Double Indexing Mechanism of Search Engine based on Campus Net", Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06).

[10] FabrizioSilvestri, RaffaelePerego and Salvatore Orlando. Assigning Document Identifiers to EnhanceCompressibility of Web Search Engines Indexes. In the proceedings of SAC, 2004.

[11] Oren Zamir and Oren Etzioni. Web Document Clustering: A feasibility demonstration. In the proceedings of SIGIR, 1998.