Enhancing Relevancy of Search Engine using Schema Matching Techniques

Sumit Jain
ATC Indore

Sanjay Tanwani
SCSIT Indore

NitikaDohan
MITM Indore

ABSTRACT

We are living in the age of global interconnectivity where Internetand related communication technology reign supreme and has taken aplace of basic necessity in business as well as daily routine. Thisinterconnectivity is enhanced extremely with the help of SearchEngine. Tools of Search Engine enable us to find information on theweb database rapidly. On the other hand this rapidity of informationavailability has not been relevant to actual user need. There have beencontinuous efforts to find relevant contents accurately but still theyare far from satisfactory. Therefore there is a need to design efficient and optimum Search Engine to surmount this problem of relevancy.

In this paper, we are proposing few novel methods to enhance SearchEngine relevancy in terms of user query. The key aim is to addressand resolve issues that crops up in search outcomes.Existing Schema matching technique identify meaningful documentand their essential features for document selection. It helps inreducing the amount of user efforts. Several matching techniques areused currently for improving the search. Still the results obtainedfrom these matching techniques are far from perfect. So in this paperwe are proposing more efficient schema matching technique namedInstance Based Schema Matching for enhancing the search result.

This Instance based schema matching is having an automatic approach to find the name of schema elements, the structure of theschema and formal ontology to improve the search outcomes byretrieving accurate data from web databases. In order to implementSchema Matching Technique for unstructured data requires WrapperGeneration process. This process is used to obtain common format ofdata from different source. It also implements a query engine whichextracts the user query relevance data from target data source. Toobtain accurate and relevant information out of the raised query thereis a need to bring result that displays the list of matched documents.

Keywords

Wrapper Generation, Query Engine, Formal Ontology

1. INTRODUCTION

The problem of accurate data extraction on World Wide Web has gained significant attention in recent years. With the advent of World Wide Web a new platform for sharing information has emerged. This has led to unprecedented level of information available at one's disposal. But shift from under availability of information to overloading of information has brought its own problem. Availability of huge database of information has also created a necessity to develop a tool for sifting and managing this information efficiently. Inrecent years, variousintelligentsmarttoolshave been developedforinformationretrieval. Thesesmarttools are able to find more userquery relevantdata thatencouragesthe listingofa particular typeofdata. Searchengineis one ofthe smarttools to explore many researchareasandto performsearchona large numberof data sources. Though it helps in retrieval of information but still it lacks in accuracy. Existing Search system has been implementedwiththree differentmodules.

In the Fig 1 shows the architecture of existing search system. Input query is fire on the query interface. It is an interface where user can express his requirements in the form of input query and this query is submitted on the web database.

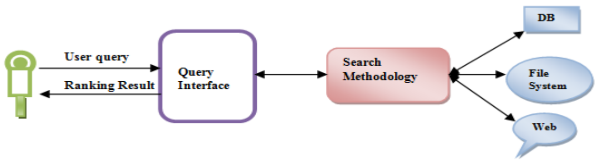


Fig-1 Existing System Architecture of Search Engine

In searchmethodology, the systemis recognizingthe inputqueryandthen after a separate methodology will perform search operation on the available data.The search results generate by the search engine which are sorted or ranked for providing the query relevant data to the end user [7][3]. On the other hand sometimes it will return a few irrelevant results that may be caused by insufficient query and semantic gap between query keywords and database available knowledge. Existing Schema Matching techniques overcome these issues by applying appropriate technique, which bridges the semantic gap between user query and database knowledge. Instance Based Schema Matching is more efficient method of Schema Matching which enhances search outcome and provides more accurate result [1].

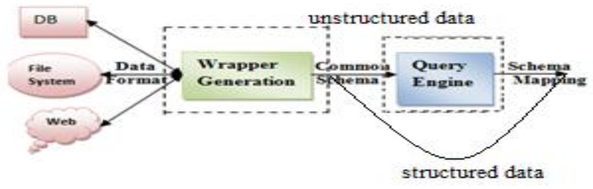


Fig 2: Proposed Architecture of SM

In this work [2][4], we proposed a novel instance-based schema matching approach is based on three observations about web database. The availability of data in web domain is found in both the formats structured as well as unstructured. Thus the entire implementation of the system is given in two major modules first for data search using the unstructured data format and using the structured formats. In this paper the data search using the unstructured data base is presented.

In the Fig 2describes how the structured data and unstructured data is processed using the proposed architecture. That includes Wrapper Generation, Query Engine and Schema Mapping. In this given system, initially user will write down input query on its query interface then the qualified input elements are located by an element identification component. After query submission on the Query interface, the result set is collected from heterogeneous format. In wrapper generation

[8], heterogeneous information from web pages is collected and then create a general model that can be recognized easily in common schema format. This common format produces as input to query engine for query optimization process. In the query engine, instance-based matchers are implemented which includes five the component such as Similarity Matcher, Tokenizer, Formal Ontology, Instance Recognition Process and Annotation Generation Process. Using all these operations, search results with semantic meaning are preserved and eliminate meaningless information. The combined outcome of the query engine will recognize with various mapping process. After mapping process, accurate search results are reported according to end user query.

Our system will use different methodology which works behind query interface in order to minimize effort of user. From this hidden methodology which works behind the interface makes searching tools very easy to learn and use. As part of our approach, system will mainly focus on detail design of wrapper generation and the query engine module for extracting relevant documents from the large databases.

The main contributions of this paper are:

- Introduction of a Wrapper Generation to generate common format from heterogeneous Web database.
- Introduction of instance-based matchers that includes five components that is added in Query Engine.
- Benefiting from the above instance-based matchers with semantic meaning and remove meaningless information.

The rest of paper is organized as follows. In section- 2 we describe Wrapper Generation. In section-3 Query Engine Concept.Section-4 Result Combination.In sections-5 Experiment Evaluation. Finally section-6 terminates with drawing conclusion.

2. WRAPPER GENERATION

Wrapper-generation is the first phase of proposed architecture of schema matching for heterogeneous Web Database [8][9]. The role of wrapper generation is as shown below in Fig 3.

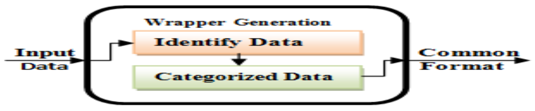


Fig 3: Role of Wrapper Generation

Wrapper Generation is an intermediate format of data where all the heterogeneous formats of data are normalized into a single common format. This common format is essentially generates for relevant information to a specific domain. Therefore, at this stage, data is manipulated and grouped according to their identification in offline environment. In other words that is a training phase of the search engine where data is processed and categorized in a specific domain labels.

In this process, all the available data **is read** first and then features are computed to recognize the data for long term, and by which the whole text or web document is represented by a small amount of data. In order to prepare the document features two basic parameters namely term frequency and sentence formation frequency is measured. For approximating these features the following formulas may helpful.

$$T_f = \frac{Totalwordcount}{totalword \in document}$$

And

$$S_f = \frac{Totalsentanceswithword}{totalsentances \in document}$$

Where the $T_f \wedge S_f$ are known as term frequency and sentence formation frequency. That can be easily understood by the below given algorithm steps in Fig 4:

```
Input: D set of documents
Output: relational domain features
Processing:
    1.  For i=0 to D.length()
    2.  String S= D[i].tostring();
    3.  String[] token= S.split(‘ ’);
    4.  String[] sen=S.split(‘.’);
    5.  totalCount= token.length()
    6.  For j=0 to token.length()
    7.  Tokencount= count(token[j])
    8.  Tf= Tokencount/ totalCount
    9.  If sen.contains(token[j])
    10. SentenceCount+1;
    11. End if
    12. End for
    13. Sf= SentenceCount/sen.lenth();
    14. Return <C, D, Tf, Sf>
```

Fig 4: Wrapper Generation Algorithm

The following diagram shows the flowchart for Wrapper Generation Algorithm in Fig 5.

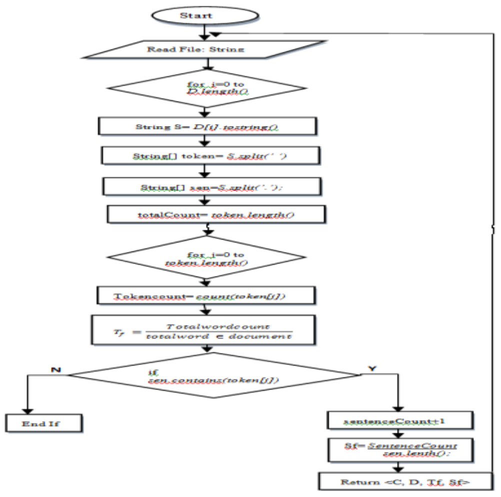

Start
Read File: String
for i=0 to D.length()
String S= D[i].tostring()
String[] token= S.split(' ')
String[] sen=S.split('.');
totalCount= token.length()
for j=0 to token.length()
Tokencount= count(token[j])
$T_f = \frac{Totalwordcount}{totalword \in document}$
if sen.contains(token[j])
N
Y
End If
sentenceCount+1
Sf= SentenceCount / sen.lenth();
Return <C, D, Tf, Sf>

Fig 5: Flowchart for Wrapper Generation Algorithm

The proposed wrapper generation algorithm is described above. In first phase, the data is read and the most common words named just like stop words (such as is, am, are, this, that) are removed. Then term frequency and sentence formation probability is estimated for unique terms of document. On the basis of estimated attributes a tuple <C, D, T_f, S_f> is calculated where C is denoted as the class name on which the word or token is found, D stands for document which is represented by the token, T_f and $S_f$are the current document’s word and sentence formation frequencies.

Now the amount of words in a web document is found in a significant amount, thus the numbers of estimated features are required to reduce first. Thus only those words are remaining that having a significant importance in the document, therefore the constructed tuple is sorted according to the T_f and $S_{f.}$ And top 50 tuples with higher $S_f$are selected.

This can be easily understood by an example suppose we have a set of document in a directory named "data structure". Thus the directory contains a number of documents of subject data structure. Thus class is data structure which is representing by C. Now during data processing stop words will be removed (such as I, we, they and other similar words). Total number of words in a document is suppose 50 and in this document the word 'Array' is occurred thrice then here the term frequency is (3/50= 0.06). Now if the document contains 20 sentences and word 'Array' found twice in a sentences, then the sentence formation probability of word 'Array' is (2/20=0.1).

3. QUERY ENGINE

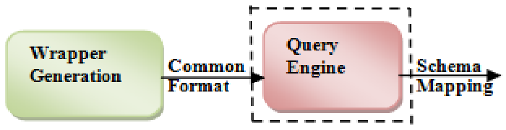


Fig 6: Role of Query Engine

In Fig 6, Query Engine is a part of proposed search engine, that accepts both ends of data, first from the wrapper of data and second from the user query to finding the correlation between query and available resultant data, in order to find the more accurate and relevant results. Therefore, required to refine and optimize the user query, because wrong or ambiguous query results the unexpected outcomes. For this purpose, we will apply instance based schema matcher which is added into query engine. Instance based schema matcher have five components, namely Similarity matcher, Tokenizer, Formal Ontology, Instance Recognition Processes and Annotation Generation Process.

3.1 Similarity Matcher

In this process, the similarity matcher is utilized to find the nearest documents from the available data features. From the nearest data two different kind of distance are evaluated namely Euclidian distance and secondly the absolute distance. Suppose there are two instances of data P and Q

Such that

$$P = p_1, p_2, p_3, \dots, p_n$$

And

$$Q = q_1, q_2, q_3, \dots, q_n$$

In order to find distance between two data instances P and Q

Euclidian distance

$$D(P,Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

And absolute distance

$$D(P,Q) = |P - Q|$$

The most relevant text features are estimated using the parsing of the user query data. Parsing is basically a process where the input text is converted in number of words. Then after using these words a number of similar strings are constructed. In order to describe Similarity Matcher is given using the below given steps in Fig 7.

```
Similarity Matcher Algorithm
Input : user query U, extracted features tuple T
Output: similar data list L
Process:
    1. Query←readInterface(U)
    2. For each feature in T
    3. Find $d(x,y)$
    4. If d(x,y)< 0.5
    5. L[i]=document
    6. End if
    7. End for
    8. Return L
```

Fig 7: Query Engine Algorithm

The following diagram shows the flowchart for Similarity Matcher Algorithm in Fig 8.

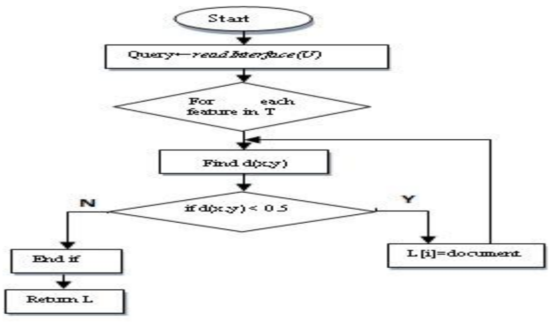


Fig 8: Flowchart for Similarity Matcher Algorithm

In the above given similarity matching algorithm first using user interface the input text is read as string this string is a set of words for instance "how to read text". This query having four words these words are compared or searched over extracted feature tuples for that purpose if all these words found in feature sets then the document can be representing able with the user query.

3.2 Tokenizer

In the next step the query is processed using the tokenizer, new sequences are created for finding the suitable query that represents user query more accurately. In order to describe Tokenizer algorithm is given using the below given steps in Fig 9.

Fig 9: TokenizerAlgorithm

```
TokenizerAlogorithm
Input : user query
Output : number of similar queries Q
Process:
    1. Query= User_query.split(" ")
    2. For i=0 to Query.length()
    3. For j=0 to Query.length()
    4. Str[]=str + query[I, j]
    5. End for
    6. End for
    7. Return Str
```

The following diagram shows the flowchart for Tokenizer Algorithm.

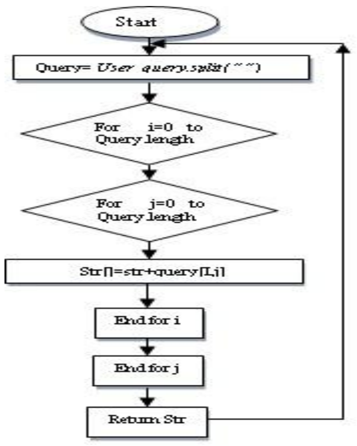


Fig 10: Flowchart for Tokenizer Algorithm

After tokenization n number of new query sequences is generated from a single instance of query. For instance user query “how to read text” the read word having different means in different aspects for instance using a code block of JAVA, C# or other kind of computer language Or the text book is written in Chinese or other language. Therefore the query is extended using synonyms and different aspects of previous search methodology. That is semantically checked and meaningless strings are removed from the ‘str’ array.

3.3 Formal Ontology

Moreover using formal ontology process the new query tokens are generated in the step [5][6]. In order to describe overall process of formal ontology algorithm is given below steps in Fig 11.

Input : string array with number of query sequences
Output : refined queries
Process: 1. For each *query* in *string[]* 2. If *is_correct(string[i])==true* 3. *Temp[i]=string[i];* 4. Else 5. *Remove(string[i])* 6. End if 7. End for 8. *Findsynonyms(querytoken)* 9. *Replaceword(querytoken)* 10. *Create a list(new generated queries)*

Fig 11: Formal Ontology Algorithm

The following diagram shows the flowchart for Formal Ontology Algorithm in Fig 12.

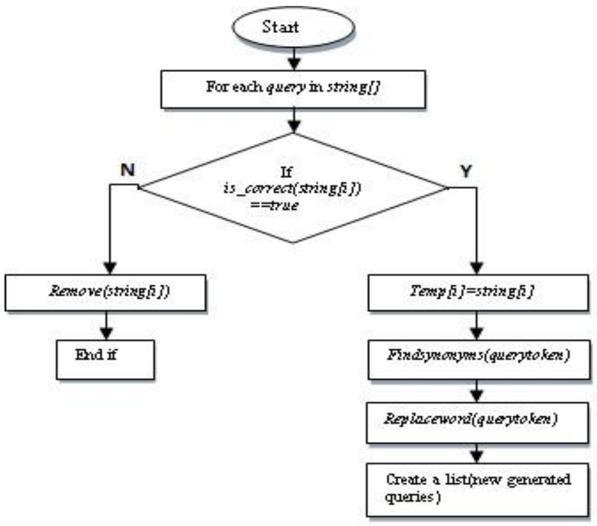


Fig 12: Flowchart for Tokenizer Algorithm

The above given process can be explained using a simple scenario suppose the user having a query “how to describe the formula of frequency paper” that can be also written as “how to define the formula of occurrence paper” or “how to designate the formulation of incidence paper” or others. In this context the last query not having a significant means semantically. Therefore this query can be removed from the list of query list using the above given algorithm steps.

3.4 Instance Reorganization Process

After processing the formal ontology steps and preparing the multiple queries from a single user query. The domain or subject of query is estimated by the prepared queries and similar class or subject of documents are explored for finding the appropriate data from the available data tuples. The process of instance reorganization process algorithm is described in the below given steps in Fig 13.

Instance Reorganization Process
Input : refined queries, document features
Output:most relevant document classes
Process: 1. *FindUniqueTokens(query_list)* 2. For each *feature* in *featurelist* 3. If *tokensexist(feature list)* 4. *ExtractClassName* and *frequency(token)* 5. End if 6. End for 7. *SortList(frequency)* 8. Find top 3 classes where query is satisfied 9. Return classNames

Fig 13: Instance Reorganization Process Algorithm

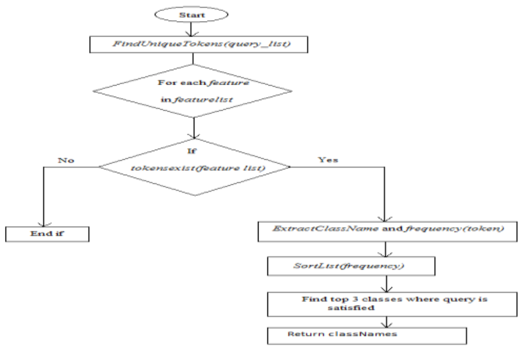

Start
FindUniqueTokens(query_list)
For each feature in featurelist
If tokensexist(feature list)
No
Yes
End if
ExtractClassName and frequency(token)
SortList(frequency)
Find top 3 classes where query is satisfied
Return classNames

Fig 14: Instance Reorganization Process Algorithm

Suppose a query "how to describe the formula of frequency paper" matched with the domain of computer science, statistics, communication, text processing and the extracted tokens of match is respectively 12, 15, 18, 3 and 19 then a sorted listed is prepared 19, 18, 15, 12, and 3 times respectively. Therefore finally the above block of algorithm returns the subjects text processing, communication and statics.

3.5 Annotation Generation Process

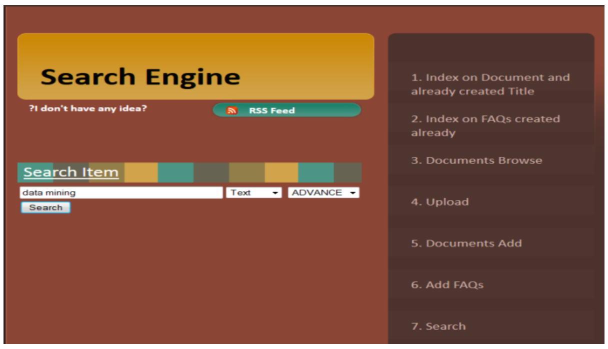
Finally the annotation for each instance of data is generated which are required to align with the similar data instances in the schema matching for data. Thus in this stage all the necessary features from the data (offline) is extracted and the user targeted information is also extracted in terms of query tokens, refined queries, and the classes which are targeted in the user query.

4. RESULT COMBINATIONS

The outcome shows the combined approach to remove wrong matching proposal and improve searching process. These combined processes are utilized for schema mapping technique to give accurate result.

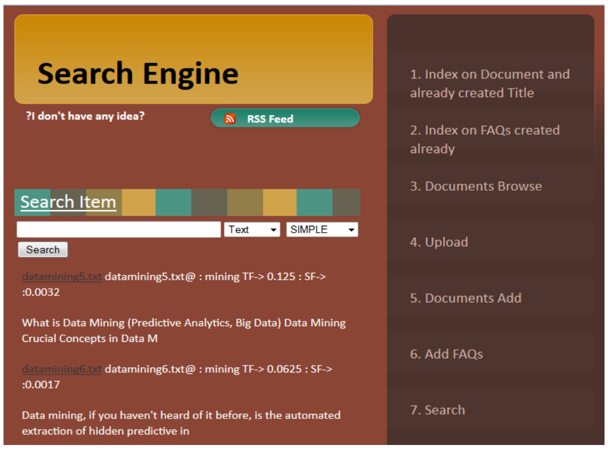
5. EXPERIMENTAL EVALUATION

We have implemented search engine using schema matching techniques. Our experiments are done on a Pentium 4 1.9 GH, 512 MB PC. The experiment is done using J2EE with HTML-based approaches. A user interface is also developed for all type of domain website where user enters his query for search. The output results are given in the form of screenshots as follows.



data mining

Fig 15: Search Page



datamining5.txt datamining5.txt@ : mining TF-> 0.125 : SF-> :0.0032

What is Data Mining (Predictive Analytics, Big Data) Data Mining Crucial Concepts in Data M

datamining6.txt datamining6.txt@ : mining TF-> 0.0625 : SF-> :0.0017

Data mining, if you haven't heard of it before, is the automated extraction of hidden predictive in

Fig 16: Search Result Page

6. CONCLUSION

This paper presented instance based Schema Matching, a system for the automatic creation of accurate result. The system uses a 2-step for accurate result. In the first step, wrapper generation process is to obtain common format of data from different source. Next is query engine, which extracts the user query relevance data from target data source. The approach are experimentally evaluated when user fire query and to obtain accurate and relevant information out of the raised query.

In the future we plan to utilize combined approach with schema mapping to remove wrong matching proposal and improve searching process.

7. REFERENCES

[1] Sumit Jain and Sanjay Tanwani.A Survey of current Research in Schema Matching Technique.In: IJCA Issue 4, Vol 3 (June 2014)

[2] Sumit Jain and Sanjay Tanwani.Schema Matching Technique for Heterogeneous Web Database.In Proc. of: ICRITO 2014

[3] Sumit Jain.Developed a General Schema Matching using Formal Ontology.CTRJ Vol II & Issue II Dec 2014.

[4] Wang, J., Wen, J. Lochovsky, F.H. and Ma, W. (2004). Instance-based schema matching for web databases by domain-specific query probing, In Proceedings of 30th Intl. Conference on Very Large Data Bases, pp. 408-419, 2004.

[5] Mario AuxilioMeina Nieto, An Overview of Ontologies (2003)

[6] Do, H.; Rahm, E.: COMA - A system for flexible combination ofschema matching approaches. In Proc. 28th Int. Conference onVLDB. Springer, 2002; pp. 610-621

[7] Z. Wu, W. Meng, V. Raghavan, C. Yu, H. He, H.Qian,R.Vuyyuru. Towards Automatic Incorporation of SearchEngines intoa Large-Scale Metasearch Engine. IEEE/WIC WI-2003 Conference,October 2003.

[8] U. Irmak and T. Suel. Interactive wrapper generation withminimal user effort. Technical Report TR-CIS-2005-02, PolytechnicUniversity, CIS Department, 2005.

[9] Hongkun Zhao, WeiyiMeng, "Fully Automatic WrapperGeneration For Search Engines",ACM,2005