ORIGINAL ARTICLES

Query Indexing Method To Improve Ontology Using Query Indexing Technique

K.Srihari, Dr.V.P.Arunachalam and Dr.S.Karthik

1Lecturer, Department of CSE, Sri Ramakrishna Institute of Technology, Coimbatore, Tamilnadu, India
2Supervisor, Principal, SNS College Of Technology, Coimbatore, Tamilnadu, India
3Joint Supervisor, Dean ,Department Of CSE, SNS College Of Technology, Coimbatore, Tamilnadu, India

ABSTRACT

Queries play a vital role in a semantic search engine. Developing a search engine involves query optimization. The queries has to be properly aligned, the queries has to be properly distributed among the database. Queries are related together and they are compared with the algorithms. This performance will be better than the query index method. Here we propose a semantic based algorithm and implemented that algorithm. A new semantic search engine is created with an URL. http://researchsrihari.replicoders.com. The results of our semantic search engine is compared to the existing search engines like hakia,swoogle,lexxe and comparison graph is generated. By comparing the result the number of quality pages visited is improved in our research. The proposed semantic search engine is practically implemented and compared with the current semantic search engines like hakia,swoogle.

Key words: Query, query index, ontology, semantic search engine

Introduction

With The Search Engine (Semantic) has an extraordinary value and demand in current web industries. It is the advanced technology which has logical relationship between the concepts (R1 ). There is a very high competition for the semantic search engines. Swoogle, Hakia, Lexxe are some of the current semantic search engines.

Problem Identification:

Existing problem:

For The existing search engines have many issues. In this research we have considered the issue of query processing in hakia search engines. Current search is done on ranking based query processing means displaying results based on hit rates. We have decided to propose a solution based on semantic search. The query processing is done and implemented using semantic search Technique. In hakia such engine query indexing is done. Here too much query is taken under consideration and those queries are processed without any standard methodology. Here no proper indexing of query is done. Some propose a standard query indexing Technique in order to implement the query traffic control and semantic query retrieval. The current search (semantic) is done and implemented to solve the queries in private data (R1). We propose a solution for query indexing in private data.

Proposed solution:

The proposed solution involves 2 main phases.
Ontology construction
Query indexing Technique.
Ontology construction: Fig.1
The general ontology is constructed based on 4 levels (or) components.
Real time applications.
Objects present in ontology
Backend-storage
Backend –storage API.

Ontology server block diagrams:

The The application generally deals with the application where ontology has to be implemented. Time / Occurrences / Situation explain the control of the occurrences of words to be searched. The notations describe the parameters (or) the representation of the word in difference word’s but of same meaning. Procedures deal with the search mechanism. Backend storage deals with the data storage. And API deals with the software compatibility. The existing ontology’s are based on the certain model where relationship is made with each other. Here we propose an architecture based on decision true.

And API deals with the software compatibility. The existing ontology’s are based on the certain model where relationship is made with each other. Here we propose an architecture based on decision true.

We propose a ontology construction technique using the decision tree based concept. The algorithm is as follows.

**Algorithms:**

Start
N ← Output nodes
Branch S ← New relationship based branch.
Branch S1 = New branch
Add new cluster (or) Tree.
All branches are given as input if no exit.
Branch output = New (branch)
Output branch = search output mode
Final result = Final output + value
All branches S to Sn are read new branches created.
Branch +0 for all N -- S
Output =New branch S2 to Sn for all sub branch S11 to Snn branch S = new sub branch(S)
For new sub branch (Single)
stop.
2) Query indexing technique generally deals with the following algorithm.
Query match (Set 1, Set 2)
Graph G = null graph (V1+V2, Σ)
Ve1 ← Set 1
Ve2 ← Set 2
(Wt1, Wt2, Wt3) ← Weights Ve1,2
For every concept X in Ve1 and Ve2 do percentage = Percentage of match (x,y) if percentage =0 or fail then add edge (f, g) to Graph G.
If (percentage = 100) or
If (percentage = cent) then
\[ W_t(x, y) = W_t1 \]
If (percentage = 80) or
If (percentage = not exact) then
\[ W_t(x, y) = W_t2 \]
If (percentage > 100) or
If (percentage = grater than exact) then
\[ W_t(x, y) = W_t3 \]
end if end for end for

Graph \( K = \text{Get match} (G) \)
If \( (K = \text{nu11}) \) then
Zero match and returns negative result.
End if,
Let \((x, y)\) denote high weight edge in \( G \).

**Experimental Results:**

**Comparison Factors:**

Here we have created a search engine url is http://researchsrihari.replicoders.com/search.php. From this search engine the following parameters are compared and improved results are obtained.
From this search engine the following parameters are compared and improved results are obtained.
Consider
\( A = \text{Proposed semantic search engine} \) \( B = \text{Hakia} \) \( C = \text{Swoogle} \) \( D = \text{Lexxe} \)

**Table 1:** Categorization parameter with Total Number of pages visited vs Total Number of Good Pages visited comparison. (Good page visited improved)

<table>
<thead>
<tr>
<th>category</th>
<th>Proposed semantic search engine</th>
<th>Swoogle</th>
<th>Hakia</th>
<th>Lexxe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicoders</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Common</td>
<td>24</td>
<td>5</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>Health</td>
<td>30</td>
<td>10</td>
<td>21</td>
<td>10</td>
</tr>
<tr>
<td>Travel</td>
<td>40</td>
<td>20</td>
<td>41</td>
<td>15</td>
</tr>
<tr>
<td>Education</td>
<td>75</td>
<td>50</td>
<td>35</td>
<td>25</td>
</tr>
</tbody>
</table>

**Table 2:** Performance of Proposed semantic search engine vs Swoogle.

<table>
<thead>
<tr>
<th>category</th>
<th>Proposed semantic search engine</th>
<th>Swoogle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicoders</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Common</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>Health</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Travel</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Education</td>
<td>75</td>
<td>50</td>
</tr>
</tbody>
</table>

**Table 3:** Performance of Proposed semantic search engine vs Hakia.

<table>
<thead>
<tr>
<th>category</th>
<th>Proposed semantic search engine</th>
<th>Hakia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicoders</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Common</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>Health</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Travel</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Education</td>
<td>75</td>
<td>50</td>
</tr>
</tbody>
</table>
Here hakia, lexxe, swoogle results are compared. The parameter taken for consideration is query indexing and our proposed solution are compared.

\[ \text{Percentage} = \frac{\text{Percentage of match (01,y)}}{\text{Returns percentage.}} \]

### Table 4: Performance of Proposed semantic search engine vs Lexxe(Good page visited improved)

<table>
<thead>
<tr>
<th>category</th>
<th>Proposed semantic search engine</th>
<th>Lexxe</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no of pages visited</td>
<td>no of good pages visited</td>
</tr>
<tr>
<td>Replicoders</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Common</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>Health</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Travel</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Education</td>
<td>75</td>
<td>50</td>
</tr>
</tbody>
</table>
After experiments we have found that if the queries are complemented with the above algorithms, we find a solution which is better than the inverted index method which is implemented in hakia and solution in for better and which can be used for public domain databases. The results are compared with the lexeme, hakia, swoogle search engines and results are display above.

**Conclusion:**

Thus we have proposed an ontology construction technique and a query processed technique. Thus two approaches have imposed in terms of indexing queries and can be implemented in public domain. In future we have decided to implement this algorithm to a larger database.

**References**


