INTELLIGENCE INFERENCE USER SATISFACTORY REFINEMENT FOR E-APPLICATIONS

P. Mohankumar, S. Prasana, G. Gunasekaran

Department of Information Technology, VIT University, Vellore, India.

Email: mkr_karai@yahoo.co.in

Received on 02-08-2016

Abstract:

In today’s world internet application relatively in all areas reached the maximum extent, amidst categories all are not significant, mostly be casual information gathering, chatting, geological searching may be discarded even results unsatisfactory, but certain aspects mandatory to process until the expected result arise. For example any online reservation, medical relevance pharmaceutical and hospital appointments, consultancies similar. This scenario we consider for our work to provide assistance for existing aspects and assuring resultant convenience for user, as research refining the prior obtained unsatisfactory result by semantically intelligence inference towards database or web applications. So that satisfying database principal implicitly. This paper provides analysis, approach, prototype, empirical test of significance for collectively data sets for the stated problem.

Keywords: Refinement, Semantic intelligence, Feedback cache, Query.

Introduction:

In today’s world internet role, we can say as human shadow such importance it become among people as well as pupils need. Though popularity as peak at level utilization, there were lot of pros and cons as surveyed in with respect to output and user satisfaction. Amidst their exist some definite application to be processed until to the level of expectation result arise at the outset. For example any online reservation, medical relevance pharmaceutical and hospital appointments, consultancies, bank transactions similar. In particular if we consider pharmacy and hospital patient reservation in real world it needs to be mandatory. If one letter or one word of a medicine was misspelled or omitted there is a possibility of lot difference among the required relevance, the entire medicine composition may get varied example anacin as anazene here head ache will become antibiotic similar phene as phenyne fever will become diaoria.
Hence it’s mandatory to process the required query exactness along with satisfactory. We concentrate on this aspect and strive to provide user a resultant related to his query be satisfactory. As a research issue we took the obtained output, tune it relevance to users unsatisfactory reason, either it may complete answer set or pending state. Here the reason received was considered as feedback related to input query-answer obtained and process it using semantic intelligence inference and event base interaction as wrapping agent to provide convenience. The road map for this paper as firstly the sample example for motivation why this in related work, following the event mode where user can view his result obtained and post his unsatisfactory reason which here considered as feedback, then secondly the storage management and issue how and where these reasons can stored and accessed during processing(tuning) that’s overall cache management\(^2\) (resultant cache, feedback cache, semantic cache) in detail, then thirdly the proposed prototype and work flows following the inference approach towards semantic intelligence based on user relevance feedback and fourthly event mapping to convenience as wrapping agent role and fifthly the empirical test for the sample dataset and test case analysis the need and novelty at the vision of significance, providing how assistance and assurance as auxiliary in bring the convenience to user answer. Finally ends with conclusion and references.

**Preliminaries and Motivation:**

Most of internets users are unsatisfactory with their application results, casual applications can be neglected but certain application must give significance until expected result arise. Here for instance we consider two samples of such cases as a motivation for our work. Case 1. Let us consider a person invokes internet for railway reservation with parameters as train no, place source, destination, date, time, if he get the desired result ok, but all time it’s not possible, instead he may get various status as waiting list, no vacancy, RAC, takkal, no berth similar etc. depending on situation, in this case he post his unsatisfaction in the event with reason, Here our system plays a vital role to resolve the problem. The intelligence\(^3\) of our system takes the previous obtained output from the system and the reason posted by user, it identifies the relevance and retrieves the related information from database, provide user the alternative possibilities exist, such as other available train status for the specified date so that the user may convinced with the provided options, if so his status can be updated. In some case these process may be repeated until user click satisfactory. The automation of this process doesn’t exist in real world application. We tested this issue for the post sort of application i.e. if the user wishes to reserve for future date for example next week or month, if he get the inconvenient result he can post his un
satisfaction in the event base option box. Later whenever he re-logs activates he will view the possible options based on his relevance which may be satisfactory. This happened because of the invoked intelligence implemented by the post tuner; here we strive, this methodology at instant base which improves the performance of system in all aspect, as well as user convenience. i.e. At the moment he receive the undesired result he can immediately post, his reason in the option provided in the event. Here at moment our concept (the autonomous intelligent agent post tuner which performs a dynamic optimal task by inferring intelligence semantically from obtained output and user posted reason for inconvenience as knowledge acquisition and mapping the event base appropriately) should get executed and display the status to the user convenience. Firstly the problem of query folding activity is analyzed. 

I.e. determining if and how a query can be answered from the given set of resources such as pervious cached results, materialized views, metadata, ontologies, as show in much are not concentrated as we emphasis intelligence from user feedback, we extract only the representation of satisfy ability, equivalence and implication relational aspects to compare with derived set of relations. Let the derived predicate S and the query predicate R under the following cases as:

1. S and R unsatisfiable, implying that the derived relation will not contribute to the answer of query R.
2. R implies S, the whole answer of R of contained in derived relation.
3. Neither S contradicts R nor R implies S, says that there may exists tuples in derived relation that contribute to the answer of query R.

any how we took the unsatisfied result only and strive for satisfiability towards intelligence as a preliminary notion we need to follow some rules which represented in below table and detailed description followed for processing as specified in Dhunam.

**Role of Cache:**

In our work we give at most importance to the cache and its utilization managements, as we deploy four caches namely, resultant cache, feedback cache, semantic cache and the query cache which will have its exclusive role as well as collective during execution in cooperative mode. Firstly we outline the issues related to the caching mechanisms then the actual concepts how the caching is utilized.

Basically caching granularity, caching coherency, cache replacement. Cache granularity used store the resultant based on the attribute, tuple and value level. Various research solutions were provided for this as specified in coming to
coherency used to ensure that the cached data are consistent with these data stored in server, further its view in the following aspects as report content updating(propagation and validation),server state(state full, stateless),dissemination mode (immediate, deferred), the cache replacement used to determine the distance to the most nearest info to the query, various algorithms resolves these issue such as WORST, WATCHMAN, CLOCK, LRU, LRU-K etc. The below tables shows the data structures table deployed in the work flow.

Table-1: User Query Log (History).

<table>
<thead>
<tr>
<th>S.no</th>
<th>User id</th>
<th>Login-time</th>
<th>Exit-time</th>
<th>Date</th>
<th>Problem-type</th>
<th>Problem-id</th>
<th>Access status (permitted/denied)</th>
<th>Result status (ok/not Ok)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10mse001</td>
<td>10:15</td>
<td>10:20</td>
<td>12/06/16</td>
<td>ATM</td>
<td>Xcz001</td>
<td>Denied</td>
<td>Not ok</td>
</tr>
<tr>
<td>2</td>
<td>10bes004</td>
<td>4:30</td>
<td>4:35</td>
<td>13/06/16</td>
<td>DD</td>
<td>XDZ001</td>
<td>Permitted</td>
<td>OK</td>
</tr>
</tbody>
</table>

Table-2: Resultant cache

<table>
<thead>
<tr>
<th>S.no</th>
<th>User id</th>
<th>Problem id</th>
<th>Output</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10mse001</td>
<td>Xcz001</td>
<td>Checked</td>
<td>Ok</td>
</tr>
<tr>
<td>2</td>
<td>10bse004</td>
<td>XDZ001</td>
<td>Checked</td>
<td>Ok</td>
</tr>
</tbody>
</table>

Table-3: Feedback cache

<table>
<thead>
<tr>
<th>S.no</th>
<th>User Id</th>
<th>Problem type</th>
<th>Status</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10mse001</td>
<td>ATM</td>
<td>Not ok</td>
<td>Money not received</td>
</tr>
<tr>
<td>2</td>
<td>10bse004</td>
<td>DD</td>
<td>Returned</td>
<td>Mistake in name</td>
</tr>
</tbody>
</table>

Table-4: Query cache.

<table>
<thead>
<tr>
<th>S.no</th>
<th>Problem Id</th>
<th>Object(attribute, value, tuple)</th>
<th>Semantics</th>
<th>Query predicate (Query-Id)</th>
<th>Query</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10mse001</td>
<td>Usrid, ATM</td>
<td>ID,Ptype</td>
<td>ATM_type</td>
<td>Q1</td>
<td>Denied</td>
</tr>
</tbody>
</table>

As shown in the above tables firstly the query log Table 3, used to store the complete history about the processed query which plays major role in order to identify the relevance to resolve the inconvenience .Following the resultant cache Table 4, which is in actual the source for the system since it provides the details of user query output along with status
and invoke him to post the reason for his unsatisfactory which is actual input for our system stored in feedback cache.

It will display the processed output based on user id and corresponding query id, whenever the user login he can view his query status based on the time or day wise, whatever the query he submitted, else instant query status also can be viewed, this is clearly discussed in the event mapping session. In Instant based application one query at a time online interaction is performed, as soon he login, based on his user id he can post a query and get his result with the information as shown in resultant cache, based on his view he can submit status ok in the event else not ok, further he may post his unsatisfactory reason in the event option provided menu box. The reason may be in the form of word, which can be viewed as keyword may be relevance to one of the relation, tuple value or attribute, or a query form or the sentence which will be take care by the semantic intelligence inference engine to identify the equivalence to process. If user wishes to view a collective status he can choose calendar basis and get the resultant status list. Any how one by one only will be processed; he can post his status by viewing corresponding query Id and post his unsatisfactory reason collectively also. The system will take this information as user feedback which stored actually in a feedback cache and assign the problem type, problem id corresponding to the query Id and processed accordingly. Later he can view the resultant (offline –post and view result later) else processed at instant base also. Then the query cache which will have the all possible query with processed output which can be accessed by query Id. Then the query cache will have the object triples semantics query predicates and corresponding outputs which play major role during intelligence inference and similarity mapping while query processing. The overall cooperation of all the cache during the work flow is shown in below diagram Fig 1, and the detailed architecture shown in diagram Fig 3, prototypes how actual work flow happens for solving the specified problem statement, is discussed clearly and the proposed algorithm further.

Figure-1. Collaborative work flow.
The above diagram shows the cooperative and collaborative work flow of the system which is actually the organizer responsibility for answering the user query in an convenient manner. Its major task is to identify the constraints based semantic definition and map to the incoming query relevance reason posted to convenience and to manage the caches which information should be retained and removed for cache optimal utility. Then to collaborate all the caches in a cooperative manner execute intelligence to extract equivalence and map to the convenience.

**Generalized Architecture:**

![General Architecture](image)

**Figure-2: General Architecture.**

The general architecture of our system is shown in the above diagram which outlines the various level layers as User interface GUI-its and event based GUI software where user can perform his applications i.e. he can choose an required environment, post his query, view results, post his reason for his inconvenience. Then the proxy agent it's the backbone of the system as well as act as a bridge between the server and user. Further it holds resultant and feedback cache which is the source for user and system i.e. user view his query answer in GUI via the resultant cache and he can post his reason for reprocess which is stored in feedback cache. Further it incorporates the query cache and query log from server to perform collaborative and cooperative work for inference intelligence to extract equivalence or exactness for convenience answer. Then the server which is actual the query processor which take overall responsibility for query execution based on information given by the intelligence agent and finally the database which stores all necessary information’s.

**Detailed Architecture:**

The detailed view of our proposed system semantic intelligence based query answer post tuner for user convenience is shown in below diagram. The block diagram is basically decomposed into two sections one database server unit and the user-agent base client unit.
(i) Database server unit.

Database server unit is responsible for query processing and data to and fro traversal storage and retrieval and it maintains the query log and the query cache, where the query log stores all the details about the user input query with status along with user id and its data structure is shown in Table 3, in above section. This is accessed during the relevance mapping by the intelligence agents similarly the query cache; this will have a details of a complete set of executed queries with results with query id which referred as problem ID by user at feedback posting level, as well this will have all the possible set of queries with answers for particular environment if the system is predefined .this will be accessed based on the query predicates with synonym defined by object triples as attribute, value or relation name, the detailed data structure is shown in Table 6.

(ii) User interface agent client unit

The client side user interface agent is responsible for interacting with user and system. It has the resultant cache, feedback cache User interface GUI and semantic intelligence inference engine. The semantic intelligence inference engine is actually the organiser of the system scenario which takes responsibility for answering user query with intelligent way and mapping convenience. The detailed role is referred as shown in 4. It undergoes the process as follows user interfacing, intelligence inference, lexical and semantic integrator, existing resource utilisation and knowledge acquisition for query matching or rewriting and execution. All the components details were explained in general.
architecture section, here we will the core concepts alone further role and responsibilities of each component will be viewed in below workflow session.

**System Workflow:**

User views his output for the submitted query in the event display which is stored in resultant cache, based on his satisfaction he can post his response as ok or not ok in the event. If status is Ok he is satisfied with the obtained output else he can click not ok and post is reason with relevance in the box provide and wait for the tuned output. His reason may be of word, sentence or query form. These reasons are collectively stored in the feedback cache. During the execution i.e. post tuning phase the semantic intelligence inference engine fetch the posted reason in feedback cache by user and identify its corresponding input query details form query log and processed result from the resultant cache using commonly the key attribute (user_id, query_id) as shown in the data structure table. Then it parses the reason, analysis lexically and identifies the synonym then using intelligence maps the relevance based on query predicates (attribute name, value or the table value), and executes i.e. the intelligence inference engine analysis all the possible and alternate query resultants based on the gathered relevance and checks for satisfiability equivalence or exactness, if so then the semantic event mapping tunes the obtained output to the user convenience In case reason of the form keyword or sentence then it checks for similarity exists in query cache if it displays the output else if user post his reason in query form the query similarity is checked in the query cache. If so then the output is fetched and displayed else the above procedure is repeated. This may be performed sometimes if user doesn’t satisfy with the obtained result.

**Proposed Algorithm:**

Procedure `POST_Tunner`.

**Input:** Query output (query_id, user_id, problem type)

// unsatisfactory to user, Problem type - UN satisfaction reason posted by user.

**Output:** satisfied output

// query answer in a user convenient manner.

**Flag** (unsatisfied = True, satisfied = False)

{ User login

If (login validity==Ok)
Displays the event GUI
Invoke the resultant table

// user views his query result in the resultant cache

Resultant Cache (user_id, query_id)

Fetch the result obtained corresponding to user_id, query_id.

If (status== ok)
{ Post another new query (); }
Else Exit;
If (status! ok)
{ Post the reason with query_id; }

Feedback cache (user_id, query_id, Problem type)

{ Update reason in feedback cache and assign problem_Id correspond to query_id. }

Invoke semantic intelligence inference engine ()

{ Fetch the feedback cache identify the form of reason as keyword, sentence or query.
If choice of reason (case)

{ Case 1: if (reason_form==keyword)

Parse the word, identify semantic relevance, and map the corresponding attribute, value and tuple relation.

Retrieve the relevance from query log and query cache.

Using semantic intelligence retrieves the similarity map to user convenience and display the output.

Case 2: if (reason_form== Sentence)

Parse the sentence word by word, analysis lexically and identify the synonym, list the relevance attribute, tuple value or relation exists.

Case 3: if (reason_form== Query)

Check the query cache whether similar or alike query exist if display as first choice output to user.

Else Submit to query processor ().}
Default: exit

}}

Query processing.

Query processor (query_id, attribute, relation, tuple value, operation).

{ //information given by semantic intelligence inference engine relation, operator, attributes value based on the cooperation and collaborative of the quad cache.

Executes the query.

  If (query== exact)

    Fetch the relevance result from query cache and submit to event mapping agent ()

  Else

    If (query ==partial)

      Process as probe and remainder query execute perform dynamic mapping and submit to event mapping agent.

    Else

      If (query ==mismatch)

        Display the invalid

      Exit. }

Event mapping.

Event mapping-_agent (user_id, query_id, output)

{// used to map the query processor output to the user convenient form.

  Receives output obtained from query processor

  Map it to the user convenient form using the event mapping software exiting as agent.

  All the relevance exact equivalence or satisfiable alternatives based on the reason posted will be displayed as answer.

  At least one in the list will be user convenient.

Let Ei be set of output satisfies equivalence and Ai be the set of alternatives relevance equivalence’s

  For Ei..En

  { for Ai.. An)
{ If(status==OK)
    Exit()  
  }
else Return;

Experimental Analysis:

In order to test our proposed work and designed system we consider to real time scenario one for instant base and other for offline base.ie one may be OLTP approach and other be of OLAP of course either or can be deployed for the two cases. We initialised already one sample as preliminary session.

Case 1. Consider a bank customer, utilise ATM machine for money transaction, all the time he may not satisfied with machine response. Some time it will respond Timed out, receipt not available, transaction complete without money dispatch, dispatched money with unspecified amount etc. User may be obviously unsatisfied with these sort results, if he wants to recover this in real time it takes minimum a week which is definitely unsatisfactory, our system provide a recovery solution for this issues .With net banking account he can post is problem cause and get solved at instant, if the server holds the feature of our system semantic intelligence inference based query output post tuner

(i) Solution by our system

User login: user_id

Resultant cache displays all the operations (transaction performed that day).

// he can filter the particular problem alone based on date and time specification.

// he can analysis his problem through the display output in more detail by invoking the query log.

He can post the reason for his problem (in the form of word, sentence or query)

(ii) Sample 1


2. Sentence base reason- ATM dispatched receipt without money today.

3. Query base – select transaction status from savings account where accno='aXXX1000' and date='15-Aug=2014'.

Where all these information’s stored in feedback cache. The semantic intelligence inference engine analyze the reason post by user in feedback cache. Semantically and fetch the relevance information from query cache, query log and find
the problem exactness using intelligence in a collaborative way. I.e. find the query in query cache relevant to the problem type and get executed by the query processor. Each cache deploys a key attribute based on this all the relevant information to the problem type i.e. the reason posted by user base is gathered. The obtained result is mapped by the event mapping agent software and display to the user in a convenient way. The sample output is shown in table 5, 6 and 7. As resultant user will receive a status towards mobile SMS or before log out the status in system.

**Table-5: Obtained Output.**

<table>
<thead>
<tr>
<th>S.No</th>
<th>User</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Login</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transaction completed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receipt dispatched</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Logout</td>
<td>unsatisfied</td>
</tr>
</tbody>
</table>

**Table-6: User Posting Reason for Post tuning.**

<table>
<thead>
<tr>
<th>User login</th>
<th>Problem type</th>
<th>Reason</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ax001235</td>
<td>ATM</td>
<td>Receipt dispatched without money</td>
<td>unsatisfied</td>
</tr>
</tbody>
</table>

**Table-7: System Modified Output.**

<table>
<thead>
<tr>
<th>Accno</th>
<th>Balance available</th>
<th>Transaction</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qx000012</td>
<td>128768</td>
<td>Last txn cancelled</td>
<td>14-Aug-2014</td>
<td>4:30 pm</td>
</tr>
</tbody>
</table>

Alert message “check account and transaction details “.

Case 2 Offline processing which can be viewed after a day.

Here we consider an academic issue for later response. In our university we offer an FFCS system for academic admin. Here the students prefer the subject, time, faculty course type, exam mode on their choice .There were lot many issue we consider one among them related to faculty work load. Initially all the faculties expected to give their convenient subject and course wish to handle ,based on the students preference all faculty may not satisfied with their workload prior a week the class work starts the work load will kept for faculty login for their updates exits if any. They can view and post their inconvenience which will be resolve based on the system possibilities. The sample is shown in table 8, 9 and 10.

(iii) Sample2.
Table 8: Obtained Output.

<table>
<thead>
<tr>
<th>User login</th>
<th>Subject</th>
<th>Location</th>
<th>Time</th>
<th>Day</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>111290</td>
<td>Algorithm</td>
<td>TT</td>
<td>8.00 AM</td>
<td>Tuesday</td>
<td>CSE</td>
</tr>
<tr>
<td></td>
<td>DS lab</td>
<td>SJT</td>
<td>9:10 AM</td>
<td>Tuesday</td>
<td>Mechanical</td>
</tr>
</tbody>
</table>

Table 9: User Reason.

<table>
<thead>
<tr>
<th>User login</th>
<th>Problem type</th>
<th>Reason</th>
<th>Status</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>111290</td>
<td>Time inconvenience</td>
<td>Continuous lab and theory hours</td>
<td>unsatisfied</td>
<td>12-Aug-2014</td>
</tr>
</tbody>
</table>

System output. Alert message “request processed”.

Table 10: Modified Output.

<table>
<thead>
<tr>
<th>User login</th>
<th>Subject</th>
<th>Location</th>
<th>Time</th>
<th>Day</th>
<th>Branch</th>
<th>date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>111290</td>
<td>Algorithm</td>
<td>TT</td>
<td>8.00 AM</td>
<td>Tuesday</td>
<td>CSE</td>
<td>14-Aug-2014</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>DS lab</td>
<td>SJT</td>
<td>9:10 AM</td>
<td>Wednesday</td>
<td>Mechanical</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graphical Representation of Result:

The graph in Fig 4 shows the comparison of the existing system with post tuner output and without post tuner. We took number of user query with unsatisfied results i.e. pending state at various time intervals. We find a significance difference at various level of time interval we observed a drastically reduction in the pending stage of resultant cache.

Post Tuner result Plot

![PostTuner response](image_url)

Figure 4: Post Tuner result Plot.
The above graph is based on the experimental analysis of existing unsatisfied result in the resultant cache with our proposed system, at the outset the unsatisfied queries are reduced to maximum considerable extent at various level of testing.

Conclusion

In this paper a novelty is proposed and proved to give the user query output at convenience level. Here we deploy a new approach of cache management for inferencing intelligence during query processing i.e. collaborating the cooperativeness of quad cache namely resultant, semantic, feedback and query cache which is deployed. As a difference we took input from the prior processed unsatisfied results and strive to user answer optimality with satisfactions. Apart various level of intelligence is involved such as to give alertness to user based on completion during offline applications towards mobile and alternate solutions in case of online application .only the sample with result comparative is noticed in this paper as its laboratory work. Thus we conclude the work as the autonomous intelligent agent post tuner, which performs a dynamic optimal task by inferring intelligence semantically from obtained output and user posted reason for inconvenience as knowledge acquisition and mapping the event base appropriately, to provide assistance for existing aspects and assuring resultant convenience for user. In overall by default satisfying the goal of database management system.

References:


7. Laurent d'Orazio* Claudia Roncancio* Cyril Labbé* Fabrice Jouanot* “ Semantic caching in large scale querying systems”

8. Dongwon Lee Wesley W. Chu “Semantic Caching via Query Matching for Web Sources “In Proc. 8th ACM Int'l Conf. on Information and Knowledge Management (CIKM), Kansas City, Missouri, USA, November, 1999


11. Heiner Stuckenschmidt Vrije Universiteit Amsterdam” Similarity-Based Query Caching. Tutorial.

**Corresponding Author:**

**P.Mohankumar**,

**Email:** mkr_karai@yahoo.co.in