Context for the Intelligent Search of Information

[1]Syopiansyah Jaya Putra [2]Ismail Khalil
[1]Department of Information System, State Islamic University Syarif Hidayatullah Jakarta, Indonesia
[2] Institute of Telecooperation, Johannes Kepler University Linz, Austria
Email: [1]: syopian@uinjkt.ac.id, [2]ismail.khlail@jku.at

***Abstract*- Three forms of contextual search have been proposed in the literature. The first one is to scan the full text of a query to figure out user needs and based on that scan, HTML pages for content will return an index of the relevant content. In this case, the user has no control over the context of the query. The second form of contextual search is used by meta-search engines and requires the user to supply explicitly contextual information about the query to increase the precision of the returned results. The meta search engine acts as a mediator between the user query and search engines. This will increase significantly the precision of the results but add to the complexity of the user interface. The third form is to automatically infer the context of the query based on the content of the other documents. This results in the modification of search results based on previous knowledge and situations. The work presented in this paper aims to develop a search engine based on a contextual search for the translation of the Quran in the Indonesian language in order to improve the performance of the search engine and provide information needed by the user based on the context of the query. In this paper, we present and discuss an algorithm that makes use of the semantics of an information source in the form of context to support the intelligent search for information over the Web or database.**

Keywords: ***Context, intelligence search, contextual search, search engine***

I. INTRODUCTION

The word context is derived from Latin for "weave together." In common usage, it refers to the parts of a written or spoken statement that precede or follow a specific word or passage, usually influencing its meaning or effect. Many words and phrases have multiple possible meanings that are clarified by the context of their use. For example, the word "bank" has several meanings: river bank, financial institution and others. Such ambiguity is rarely a problem in common practice because content makes the meaning clear: "she is walking along the bank," "I paid the money at the bank."
Context as the associations between a central item of interest and surrounding items. The existence of the surrounding items clarifies the "meaning" of the central item. Typically, in information services, context is coded as metadata describing both the existence of a relationship between two items and the nature of the relationship."
Two strategies have been advocated for the design and implementation of contextual search: Global As View (GAV) and Local As View (LAV).

The (GAV) follows the traditional strategy developed for federated databases [1]. The global view is constructed by several layers of views on the relations exported by the sources. Queries are expressed regarding the global view and are evaluated in a conventional way [2].

The (LAV) considers that the relations exported by the sources are materialized views defined on virtual relations in the global schema. Queries are still expressed regarding the global schema. To evaluate a query, a rewriting regarding the component schemas needs to be found: this process is called Answering Queries Using Views (AQUV) [2].

The GAV and the LAV strategies [3] can be qualitatively or quantitatively compared in terms of their adequacy (1) to model a particular integration situation, (2) to cope with autonomy of the sources (sources changing their exported schemas, joining or leaving the network), and their ability (3) to answer queries.

The main arguments against the GAV strategy are that (1) it may not be able to model the integration situations where sources are missing to build the complete world view; (2) it may stop to offer a complete global view as some sources become unavailable or services are disrupted [4]. In favor of GAV, it can be argued that most practical applications will require sufficiently simple global schemas (unions) to avoid such difficulties and that there might be enough economic incentives for participating in the network to convince the sources' managers to play the game. The strength of GAV is that, if the modeling is successful, (3) all queries on global schemas are guaranteed to be answered, and a complete answer can be constructed.

The LAV strategy, conversely [4], is designed to cope with a dynamic, possibly incomplete set of sources. The counterpart of this flexibility is that all queries may not be answered or only an incomplete answer can be found. It can be argued for LAV that to great information infrastructures such as the WWW, complete answers are rarely expected or needed by the users: "better some answers than no answer". Additional semantic knowledge in the form of "context" can be taken into account in the GAV strategy to optimize the queries to the component sources. For instance, a powerful optimization consists in identifying sources that cannot produce results participating in answer to a given query or sources that would produce redundant answers. As we have discussed and illustrated in [5], there are many very concrete examples where simple context knowledge about component sources can help generating more efficient query plans. The optimization process involved is called Semantic Query Optimization (SQO).

In this paper, we argue that there is a relative duality between LAV and GAV on their use of semantic knowledge expressed in the form of context. We present and discuss a variant of the algorithm for Answering Queries Using Views

(AQUV) for the LAV strategy regarding Semantic Query Optimization (SQO). We call the process of rewriting the query using semantic knowledge Contextual Search (CS).

II. LITERATURE REVIEW

Contextual search refers to proactively capturing the information need of a user by automatically augmenting the user query with information extracted from the search context; for example, by using terms from the web page the user is currently browsing or a file the user is currently editing [6]
Implementing contextual search involves two tasks. The first is the user interface, which has been extensively discussed within the context of Y!Q [7]. The second consists of extracting and representing the context, and using the context in conjunction with the query

Search engines, generally, return results without any regard for the concepts in which the user is interested [10]. Context Search is different from Keyword Search because Context Search consider user's context when searching, meanwhile, keyword Search just care keyword matching without taking user's context into account. Context Search is based on user's context which include user's time, location, input, needs, habits and background, therefore the information about user's interest are to be collected before search, so that Context Search can understand users' search intention better [11]

The ability to promote and re-purpose content in different contexts along with the facility to suggest and find related things that weren't directly searched creates a means to surface seasonal, campaign related or tactically important information. [9].

Context is additional information about user's interest. Context will be represented by complex context vector, which is a vector of vectors. It will have number of dimensions equal to the number of document attributes used for search. Every dimension will represent one attribute and contains a ranked vector.[11]

Contextual search using ontology-based user profiles [11] consist of the following four steps. First, to represent information and knowledge in different domain, Domain Ontology is used to annotate domain information which is the basis of information representation in Context Search, Second, to extend the keyword input, with the help of Context Ontology the keyword inputted by users will be extended to Search Ontology which normalizes user's search information and context information this time, Third, search results are represented using Faceted Ontology which is also a kind of Domain Ontology for annotating the target documents, and Finally, the core of Context Search is to implement ontology matching. The method is to filter sub-ontology which is the most similar with Search Ontology from Faceted Ontology.

Methods for searching with context could be classified as [11,10]:

- *Query rewriting* – appends keywords form context to search query as a string and sends it to standard search engine. Aaunt's each query with appropriate terms from the search context and uses an off-the-shelf web search engine to answer this augmented query [6]
- *Iterative-Filtering meta-Search* – generates many different sub-queries and sends them to multiple search engines. Results are re-ranked and aggregated into one set. [9,8], generates multiple sub queries based on the user query and appropriate terms from the search context, uses an off-the-shelf search engine to answer these sub queries, and re-ranks the results of the sub queries using rank aggregation methods [6].
- *Rank-biasing* – Query and context of keywords are sent to modified search engine as complex query. At first, method finds documents matching query and then re-ranks them by fitness to context [11] [6], generates a representation of the context and answers queries using a custom-built search engine that exploits this representation [6].
- *Flooding algorithm* to match Search Ontology with faceted Ontology [9]
- *Inconsistency Ontology Reasoning Framework*

A. LAV *as a Search Strategy*

For our purpose, the schema of a relation is solely defined by the name of the relation and its arity (number of attributes). As we have seen, LAV models the integration situation as a database (schema) DB = (S_{EDB} , $S_{IDB,}$ *V, C)* where S_{EDB} is the global schema, S_{IDB} is the union of the local schemas, *V* is the set of view definitions for the component relations in terms of the global relations, and *C* is a set of contexts on the component schemas.

Despite the reality of the situation in which the global schema is a set of virtual relations and the local schema is a set of materialized relations, DB = (S_{EDB} , $S_{IDB,}$ *V,C)* stands for a database in which:

- S_{EDB} is the set of schemas of the relations in the extensional ddatabase, i.e. the set of stored relations,
- S_{IDB} is the set of schemas of the relations in the intentional database, i.e. the set of relations defined by means of views (we call both the relation and its definition a *view*, when there is no ambiguity
- *V* is the set of view definitions in Datalog (range-restricted function free Horn clauses),
- *IC* is the set of contexts in Datalog$^{\exists}$ (i.e. we allow variables occurring only in the head of the clause to be existentially quantified after all other variables are universally quantified as usual).

Queries are views, which are not defined a priori in the database schema

In this and the following section, we concentrate on the simplest problem of Project-Select-Join (or PSJ for short) queries and PSJ-view definitions. Therefore we consider an initial database *DB* such that view definitions are conjunctive, i.e. views defined by means of a single Horn clause. We also

consider conjunctive queries only: the single Horn clause defining the query is also restricted to contain literals from the extensional database in its body.

Given an instance of the extensional database *I*, i.e. a set of tuples for the relation $S_{EDB,}$ an instance of the database is defined by the minimal model of V $\cup$ *I*. An instance of the database is consistent if it is a model of *IC*. In our case, we only need to verify that *I* is also a model of *IC*. Since the views are materialized only, say *J* is the actual instance of the intentional database, an instance of the database is an instance $I \cup J$ such that it corresponds to the minimal model of $I \cup V$, and *I* is consistent with *C*. Notice that *I* may not be unique. For the sake of simplicity, we will assume that the designer of the integration solution has been careful and that a minimal instance is guaranteed to exist.

B.A Dual View of the Database

In order to comply with the reality of the situation in the modeling, we would need to construct a database DB" = (S_{IDB}, $S_{EDB,}$ *V", C")* with the same instances as *DB*. In [12] a polynomial time algorithm is given which constructs similar database (DB"' = ($S_{IDB,}$ $S_{EDB,}$ *V"', 0)*). However, it is also shown that, in the general case, the new database is only maximally contained in the initial database $DB'' = \subset_{max} DB$. In standard database terms, any set of views may not be a lossless decomposition.

Using similar transformations like the one used in [4], we construct a database DB' = ($S_{IDB} \cup S_{EDB,}$ *0, 0,C')* which has the same instances as *DB*.

To construct *C'* we use Clark's completion (see [13]) of the view definitions *V*. In this way we explicitly express the Closed World Assumption. More precisely, given a view definition for *v*:

- $v(\bar{X}) \leftarrow B(\bar{Y})$

we first make all the implicit quantifications explicit:

- $\forall \bar{X}\, v(\bar{X}) \leftarrow \exists(\bar{Y} - \bar{X}), B(\bar{Y})$

the completed axiom is

- $v(\bar{X}) \leftrightarrow \exists(\bar{Y} - \bar{X}), B(\bar{Y})$

which we transform into two integrity constraints:

- *(1)* $v(\bar{X}) \rightarrow \exists(\bar{Y} - \bar{X}), B(\bar{Y})$
- *(2)* $B(\bar{Y}) \rightarrow v(\bar{X})$

Notice that we keep a conjunction of literals in the right-hand side of the first constraint instead of separating it in as many contexts in the Horn clause as there are literals in $B(\bar{Y})$.

If we call $\hat{V}$ the set of contexts of type (2), the new database $DB' = (S_{EDB,} \cup S_{IDB,} 0,0,V \cup C \cup \hat{V})$.

We claim but do not prove that the original and transformed databases have the same instances.

Let us now look at the application of the transformation on an example.

Example 1 *Let us consider the following database schema:*

- *DB = ({w/3}, {v/2}, {v(X,Y)* $\leftarrow$ *p(X,Y,Z)}, 0)*

The view v/2 is the projection of the first two attributes of the global relation w/3. According to the transformation of the database we have described, we generate the following two contexts:

- $i_1 : p(X,Y,Z) \rightarrow v(X,Y)$
- $i_2 : v(X,Y) \rightarrow \exists Z, p(X,Y,Z)$

we are now considering the database schema:

- $DB' = (\{w/3, v/2\},0, 0, \{ i_1 , i_2 \})$

Example 2 *According to the transformation of the database we have described, we generate the following four contexts:*

- $i_1 : w(T, I, A,S, P) \rightarrow c(T, A, S)$
- $i_2 : c(T, A, S) \rightarrow \exists I,P, w(T, I, A, S, P)$
- $i_3 : w(T, I, A,S, P) \rightarrow s(I, T, P)$
- $i_4 : s(I, T, P) \rightarrow \exists A,S, w(T, I, A,S, P)$

We are now considering the database schema:

$DB' = (\{w/5, v/3\},0, 0, \{ i_1 , i_2, i_3 , i_4 , i_5\})$

Where $i_5 : s(I, T,P_1), s(I, T, P_2) \rightarrow P_1 = P_2$

C. An Algorithm For Contextual Search

Since the two databases have the same instances, we can now define AQUV as a specific SQO process: we want to transform the query into a semantically equivalent query, which uses literals from the initial intentional database or built-in literals only. Indeed, the algorithm we propose generates strong folding; although it could easily be modified to produce partial folding, we have restricted its output to complete folding.

The algorithm is similar to the one in [14]. However, we highlight the relationship between this transformation and semantic query optimization as described in [12] and [13]. Given a query of the form $q(\overline{X}) \leftarrow l_1, \ldots, l_n$, for each constraint whose body subsumes a subset of the body of the query, we call the head of the constraint after matching a *residue*. For each residue it is possible:

- either to add it to the query [*query expansion* or *join-introduction*]
- alternatively, to eliminate a literal l_i in the body of the query provided that certain conditions be fulfilled [*query contraction* or *join-elimination*]

According to [15], *Join elimination is possible if [the residue] r can be resolved [unified] with any of the* $l_{i,}$ $\exists\sigma \mid \sigma(r) = \sigma(l_i)$*, and the resulting resolved, after elimination, contains both [all] the original terms in all* l_j , $j \neq i$. This definition trivially extends to the case where the residue is of form $r_{1,\ldots}$, $r_n \leftarrow$ and a set of literals l_i such that a subset of the literals in the residue can be resolved (unified) with the set of literals $l_{i,}$ in the body of the clause. The same conditions as above apply to the resulting query.

The correctness of the join-introduction and join-elimination is proved in [13]: the *semantic query transformation* leads to *semantically equivalent queries.*

The principle of the algorithm is to use the rules of type (2) (subsection 3.1) for expansion and the rules of type (1) for contraction.

D. Algorithm and Implementation

After the generation of the sets of integrity constraints used for join-introduction (I-Rule) and join-elimination (E-Rule), respectively, the algorithm can be summarized as done in Figure 1.

The implementation of this algorithm in Prolog can be done in two different ways. Views, queries, and constraints can be represented by ground expressions, in which case some special notation is used to represent the variables in the query, and unification, sub assumption, or variance need to be written from scratch. Alternatively, the variables, terms, and constants in the views, queries, and constraints can be represented by variables, terms, and constants of Prolog. In this case, Prolog Unification as well as the built-in or library predicates for (term-) subsumption and variance can be used. Term decomposition using = ../2 or variables/. I support can not be avoided (e.g. find the distinguished variables). The latter implementation is not necessarily simpler as it hides some important aspects of the implementation and runs the risk for the programmer to inappropriately unify terms. The rules and procedures for join-introduction and join-elimination resemble the Constraint Handling Rules. It is possible indeed to use an approach similar to the one we advocated in [12] in order to implement the above described algorithm.

```
Input:
        Q: query
        I-Rule: set of contexts
        E-Rule: set of context
Output:
        Q : transformed query
for each R ∈ I-Rule
        for each σ such that σ(Body( R )) ⊂ Body (Q)

        I := I ∪ Head(σ( R ))
        end for
end for

for each L ∈ I with predicate P
        for each R in E-Rule(P)
                if ∃σ such that σ (Body( R )) = L
                then remove any literal
                matching Head (σ( R )) from Body
```

Figure 1: CS algorithm

VI. Conclusion and Future Work

In this paper, we presented and discussed an algorithm that makes use of the semantics of an information source in the form of context to support the intelligent search for information over the Web or a database. This final goal of the project from which this paper stemmed is to develop a search engine based on a contextual search for the translation of the Quran in the Indonesian language to improve the performance of the search engine and provide information needed by the user based on the context of the query. One approach in conducting a contextual search for the Indonesian translation of the Quran is to use ontologies to map relationships between documents so that it can do a query against a specific keyword to the overall structure of an existing document. The methodology of a Search engine with a contextual design based on Rank Biasing is as follow preprocessing, a query with context.

References

[1] Sheath and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Database. ACM Computing Surveys 1990.

[2] Ibrahim, I. K., Winiwarter, W., & Bressan, S. (2001). Rewriting Rules for Semantic Query Transformation in E-commerce Applications. In *Knowledge Management And Intelligent Enterprises* (pp. 130-143).

[3] Bressan, S. , Ibrahim, I, Soesioantto. (2002).IKA: Unity in Heterogeneity. International Conference and Web-based Application & Services.

[4] Duschka and M. Genesereth. Answering Queries Using Recursive Views. In Principles Of Database Systems (PODS), 1997.

[5] Khalil, I., Semantic Query Transformation for the Intelligent Integration of Information Sources, Ph.D., thesis, Gadjah Mada University, 2001.

[6] Kraft, R., Chang, C. C., Maghoul, F., & Kumar, R. (2006, May). Searching for context. In Proceedings of the 15th International Conference on World Wide Web (pp. 477-486). ACM

[7] Greenly, W., Sandeman-Craik, C., Otero, Y., & Streit, J. (2011). Case Study: Contextual Search for Volkswagen and the Automotive Industry. Tribal DDB UK.

[8] Challam, V., Gauch, S., & Chandramouli, A. (2007, May). Contextual search using ontology-based user profiles. In Large-Scale Semantic Access to Content (Text, Image, Video, and Sound) (pp. 612-617). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE

[9] Zhuang, Z., & Min, W. (2012, October). Context Search Based on Inconsistent Ontology Reasoning. In Semantics, Knowledge and Grids (SKG), 2012 Eighth International Conference on (pp. 185-188). IEEE

[10] Navrat, P., Taraba, T., Ezzeddine, A. B., & Chuda, D. (2008, September). Context search enhanced by readability index. In IFIP International Conference on Artificial Intelligence in Theory and Practice (pp. 373-382). Springer US.

[11] Navrat, P., & Taraba, T. (2007, November). Context search. In Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops (pp. 99-102). IEEE Computer Society.

[12] S. Chakravarthy, J. Grant, and J. Minker. Logic Based Approach to Semantic Query Optimization. ACM Transactions on Database Systems, 1990.

[13] J. Minker. Foundation of Deductive Databases and Logic Programming. Morgan Kaufmann, 1987.

[14] Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering Queries Using Views. In Proc. Of the ACM Conf. On Principles of Database Systems (PODS), 1995.

[15] R. Ramakrishnan and A. Silberschatz. Scalable Integration of Data Collection on the Web. Technical report, University of Wisconsin-Madison, 1998.