Application of a Semantic Search Algorithm to Semi-Automatic GUI Generation

Maria Teresa Pazienza[1], Noemi Scarpato[2,3], and Armando Stellato[1]

ART Group,
[1]Dept. of Enterprise Engineering
[2]Dept. of Computer Science, Systems and Production

University of Rome, Tor Vergata
Via del Politecnico 1, 00133 Rome, Italy

[3]University Telematica San Raffaele Roma
Via Val Cannuta 247, 00166 Rome, Italy

{pazienza, scarpato, stellato}@info.uniroma2.it

Abstract

The Semantic Search research field aims to query metadata and to identify relevant subgraphs. While in traditional search engines queries are composed by lists of keywords connected through boolean operators, Semantic Search instead, requires the submission of semantic queries that are structured as a graph of concepts, entities and relations. Submission of this graph is however not trivial as while a list of keywords of interest can be provided by any user, the formulation of semantic queries is not easy as well.
One of the main challenges of RDF Browsers lies in the implementation of interfaces that allow the common user to submit semantic queries by hiding their complexity. Furthermore a good semantic search algorithm is not enough to fullfil user needs, it is worthwhile to implement visualization methods which can support users in intuitively understanding why and how the results were retrieved. In this paper we present a novel solution to query RDF datasets and to browse the results of the queries in an appealing manner.

Keywords: Semantic Search, Web Page Visual Analysis, Semantic Web.

1. Introduction

The term Semantic Search has been traditionally adopted with two meanings, according to the interpretation of these words in different research communities: on the one hand, it has been a common expression for what is technically known as Semantic-driven IR, the application of semantic technologies to the traditional IR problem.
In (Mangold, 2007) Semantic Search is defined as a document retrieval process that takes advantage of analyzed domain ontologies to understand the meaning of keywords and discover relations among them.
The second interpretation relates to Semantic Web Data Search, which mainly deals with the retrieval of semantic web data. Many approaches following this second interpretation have been published in these last years, embracing several application areas and exposing different realizations.
In (Tran, Cimiano, Rudolph, & Studer, 2007) the authors try to map keywords of queries to ontology concepts; this is very important to understand users information needs and refine queries to get more precise results.
In (Lei, Uren, & Motta, 2006) the authors presents a semantic-based keyword search engine, the system propose a simple query interface that hides complexity of semantic query to the users. Recently, answering keyword-based queries on graph-structured data has emerged as an important research topic: an algorithm for the exploration of top-k matching subgraphs has been presented in (Thanh Tran, 2009).
Retrieval of information in a web page is a more difficult task respect to the traditional document search, because a web page is not a good information unit in which to search information.
One of the main problems of the traditional information retrieval in web pages, is that often they contain multiple topics and/or many irrelevant information as example navigation structures, decorations, and interaction part of the page.
The individuation of the informative blocks that correspond to different topics in a web page should be a good solution to solve the multi topic (Wu, 2006).
The web page segmentation can be used to make the identification of blocks of a web page and to discriminate between informative and non-informative blocks.
Following this idea, search providers have implemented several algorithms able to scan the visual content in which a page is organized to find the structure of the web content. In this paper it is adopted the idea that a web page visual analysis algorithm should be provided before the application of the semantic search algorithm to improve the performance of the latter.
However a good semantic search algorithm is not enough

to fulfill user needs; it is worthwhile to implement some visualization methods which can help users instantly and to intuitively understand why and how the results are being retrieved.

In (Myungjin Lee, 2010) Myungjin Lee presents a semantic association-based algorithm and shows how to provide proper visualization and navigation methods for the results. Dadzie & Rowe in their survey in (Dadzie & Rowe, 2011) describe many different approaches to visualize Linked Data, underlining the massive grow up of research on linked data visualization.

In (Pazienza, Scarpato, & Stellato, 2010) we defined our approach to automatic generation of GUI for browsing Semantic Web data.

In this paper we present SAGG, a semantic search algorithm to support "live generation" of GUIs from examples in web pages selected by the user.

The web is full of applied data visualization patterns: tables, forms, item lists, enhanced paragraph formatting, be them interactive (i.e. User Interfaces) or not, all provide interesting examples on how to represent the information coming from knowledge and data repositories. Our idea is to apply the basic concept of information reuse, to allow non-developers to be able to easily generate visual components for their own data, and the code necessary to populate them, by presenting concrete examples taken from a similar or identical domain. These pseudo-generated components can then be refined, but with a large save in development effort and a sensible enhancement in terms of rapid-application prototyping.

In this paper we introduced more in detail the concept idea behind SAGG, also we described the algorithm for inducting visualization patterns from selected graphical objects; finally we presented a scenario for the application of SAGG and evaluated the obtained results.

2. Overview

SAGG foresees a seed composed of a series of sample documents[1] representing information related to the same domain of (or with strong overlap with) the user's data for which a UI is to be generated.

These documents are analyzed to identify representation patterns, that is how the information is structured and presented into the web page, and how this structure could be populated generating similar results, by querying the data owned by the user. In this sense, this approach "wraps up" previous research work on Keyword-based Semantic Search With, such as (Lei, Uren, & Motta, 2006), with two non-trivial extensions: by first, keywords are extracted from the content of the sample documents instead of getting them from the user; secondly, the output is not a set of triples but a reverse-engineered query able to re-extract the same data of the samples from the user dataset, in order to populate the sample forms. This algorithm thus enables an easy "create-by-example" approach to generation of user interfaces and queries.

2.1. Model

The data model behind SAGG algorithm foresees a set of elements which are identified and analyzed over standard documents. We adopt the DOM (Le Hégaret, Whitmer, & Wood, 2009) standard structure to make assumptions regarding content of the page. We call the input of GUI Generator *PatternExample*; a *PatternExample* is an HTML page or part of this that users identify as relevant for their domain. A *PatternExample* can be segmented into a set of recognizable Graphic Objects (GOs):

$$PatternExample = \{GO_0, GO_1, \ldots.. GO_n\} \quad (1)$$

A GO is a part (presentation & content) of a *PatternExample* exposing relevant (and mostly self-contained) information. SAGG assumes content of GOs as independent information units i.e. information which can be analyzed separately from the rest of the page with respect to the user data. As from the adopted DOM formalism, the presentation of a GO can be structured through a tree with a root element (e.g. TABLE, LIST ...) identifying its nature, and a list of tree children.

In each GO we can recognize a list of atomic information units called GE.

$$GO = \{GE_0, GE_1, \ldots.. GE_n\} \quad (2)$$

In our approach we assume that a pair of GEs contained in the same GO may be bound through a relationship. We have defined the following grammar to identify the different relations which can hold between.

$$(GE, +, \times) \quad (3)$$

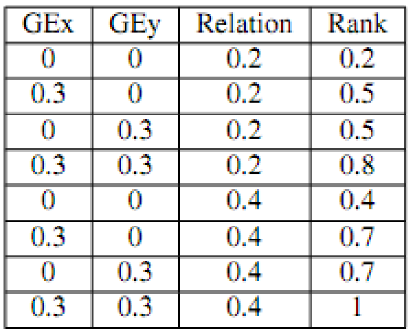
In our grammar we have identified two types of relations:

- + being sibling
- x dependency

The sibling relation is a generic relation that is established between pairs of GE which are part of the same GO and are considered as peers with respect to the content analysis process[2].

[1] We often refer to examples taken from web pages and thus implicitly refer to HTML components, though the presented algorithm can be applied to any common document model, such as the generic XLS-FO (http://www.w3.org/TR/xsl/) model.

[2] The choice of the name "sibling" is appropriate with respect to the semantic analysis and assessing of relationships among GEs. On a syntactical perspective, this should not bring confusion with the fact that "sibling" elements may actually be nested on each other or belong to different branches of the DOM tree and thus not be sibling in the hierarchical organization of elements.



GEx	GEy	Relation	Rank
0	0	0.2	0.2
0.3	0	0.2	0.5
0	0.3	0.2	0.5
0.3	0.3	0.2	0.8
0	0	0.4	0.4
0.3	0	0.4	0.7
0	0.3	0.4	0.7
0.3	0.3	0.4	1

Figure 1: Ranking Value

A further – more specific – relation, called *dependency*, can be established among pairs of elements where one of them acts as a pivot for the other (the relation has thus a verse).
Usually a pivot establishes a dependency with several other elements (e.g. a column header in a table is a pivot for the content of the cells in its same column).
This type of relation implies a stronger binding with respect to the sibling relation (which relies on basic keyword-based search solutions over graphs), so that the algorithm can apply more constraints on the search.
We thus define for each GO a list of triples:

$$<\mathrm{GE}_x\ ,\mathrm{relation},\mathrm{GE}_y> \qquad (4)$$

These triples are ranked according to the following method.
First of all, a weight for the sibling relation (0.2) and for the dependency relation (0.4) have been empirically assigned . Then to determinate the weight for each GE involved in the relation, an algorithm to check if some GEs can be mapped with some to RDF nodes of in RDF Schema the dataset has been implemented. If this condition is verified a weight equal to 0.3 is assigned to it. The fiond rank of a triple is the sum of the weight of its relation and weights of each of the two GEs.
In the Figure 1 all possible combinations of values of weights for a triple are shown.

2.2. Overview of the algorithm

In this section we make an overview of the different steps involved in the SAGG process. The two main parts of the algorithm are related to the induction of representation patterns and the estimation of the queries which can populate them in a way similar to the presented example, but taking data from the user's own dataset.
These part are implemented by the introduction of the the SAGG's VIPS algorithm and the SAGG's SSA algorithm.
The key idea behind our algorithm is to analyze information units (GOs) instead of processing all the text from the input page. Another key aspect is that we use the structure of information units to identify atomic information (GE) and to understand their peculiarity. In particular the relative position of GEs in a recognized GO is meaningful to establish relationships between GEs and to assess the nature of these relationships. Here below, we detail the different steps of the algorithm:

2.2.1. Pattern Recognition

In order to identify graphic objects (GO) and their graphic element (GE) we defined a rule-based webpage segmentation algorithm to divide the *PatternExample* into GO information units.
Once the GOs have been recognized, we analyze each GO to identify its GEs and generate a list of triples representing relationships between them.

$$<\mathrm{GE}_x\ ,\mathrm{relation},\mathrm{GE}_y> \qquad (5)$$

Depending on the nature of the GO (e.g. list, table, etc..) we have implemented customized strategies with different assumptions regarding the establishment of relationships among GEs. The result of this step is thus a list of GOs and their associated list of triples over GEs and relations.
Many details about this algorithm, called SAGG's VIPS algorithm, will be provided in the next section.

2.2.1.1. SAGG's VIPS algorithm

In this paper it is applied the idea that a web page visual analysis algorithm should be provided before the application of the semantic search algorithm to improve the performance of the latter. Indeed a VIPS algorithm has been developed to recognize relevant blocks (called GO from now on) in a web page.
The key idea behind SAGG's VIPS algorithm is that the position of keywords in an html structure is meaningful to understand the relations between the keywords.
SAGG's VIPS algorithm defines a chain of actions to identify relations between GEs .
For a set of special kind of tags (e.g. title, table) some customized strategies has been implemented, to make further assumptions regarding the nature of relations and to identify relations of dependency between particular kinds of GEs.
To realize SAGG's VIPS algorithm first of all a grammar has been defined. The Definition of this grammar makes possible the formal description of the relations that exist between data that are inside of different tags in an HTML page.
The inputs of SAGG's VIPS algorithm are : a web page (or a portion of the web page) identify by the users in the web; and a knowledge base indicated by the users itself.
The main assumption of SAGG's VIPS Algorithm is that the web page is chosen for users because it is a good example of visualization pattern for the semantic data stored in the knowledge base. This means that users recognize into the page both a desired graphical representation and data that are in the same domain of them.
Considering that the goal of SAGG is the creation of a new GUI, in the SAGG's VIPS algorithm a complete mapping between DOM tree of page and visual tree has

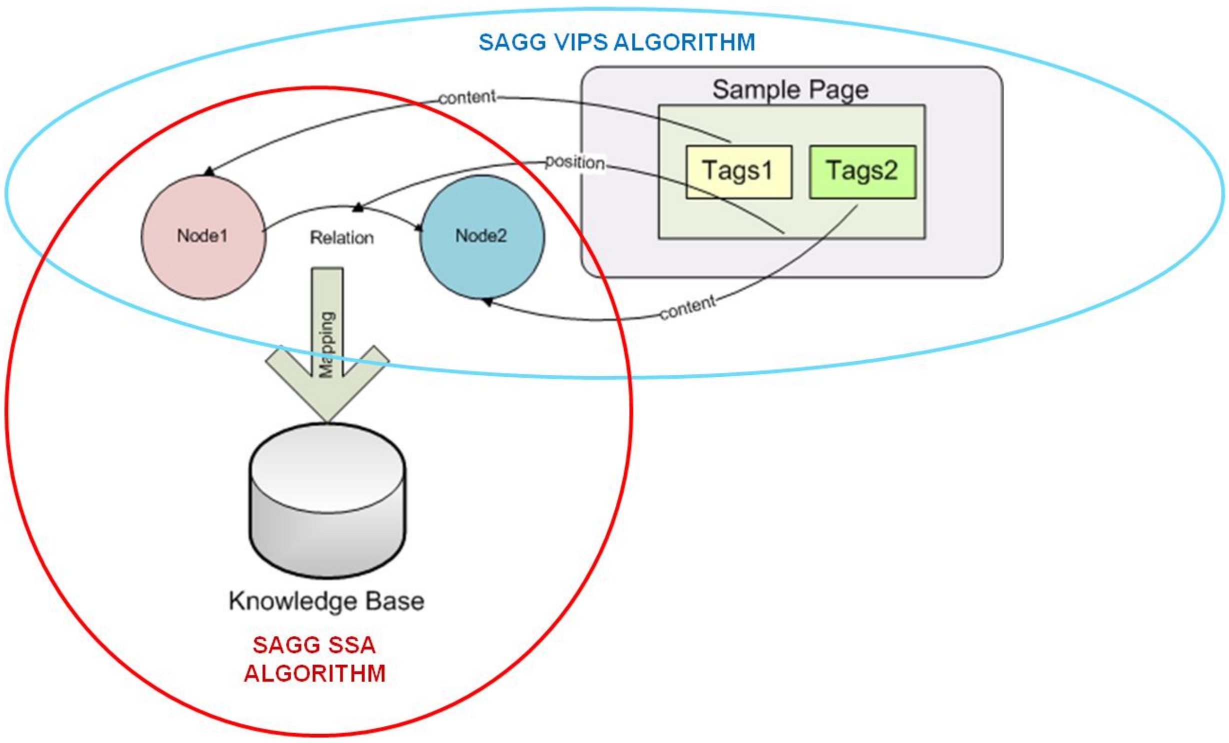


Figure 2: Interaction between SAGG's SSA Algorithm and SAGG's VIPS Algoritm

been implemented with the aim to reproduce the same page structure, in the created GUI. In our approach a GO is characterized by a root element that correspond to a DOM node and by a list of GEs, each of this GEs ,in turn, is characterized with a DOM node that is a child node of the root node related to the GO.

In the visual tree provided by In SAGG's VIPS Algorithm it is possible that, some GO has as root element a DOM element that is a child of another GO root element. In this case the first GO it is composed only by the portion of html that surround the second GO that is pulled out from the GO . This because in the SAGG's approach, it is assumed that the overlap between GOs is not permitted.

The first step of SAGG's VIPS algorithm is to recognized a list of self contained units called GO that can be analyzed separately

The second step of SAGG's VIPS algorithm is the analysis of each GO, in this phase the algorithm recognizes in each GO a list of a single unit of information (called GE from now on) and their relations.

Differently from traditional web page visual segmentation approaches where block level is the most granular level of segmentation, in the SAGG's VIPS algorithm approach it was introduced a further segmentation level. In fact in each block it is recognized a list of GEs.

In this paper it is assumed that to understand the meaning of a GO it is useful identify a list of GEs that compose it. After the identification of the GEs it is also defined a graph of relations between GEs analyzing the position of each GE into the GO.

To recognize a GO some rules have been defined, these rules are related both to visual and structural features.

Furthermore a rule that allow to recognize and discard not meaningful elements (e.g. comment or script that are not interesting for SAGG purpose) has been defined.

The realization of this rule is achieved through the definition of a stop list of non-relevant elements.

Same as GOs to recognize GEs a set of rules has been defined. In this approach it is assumed that the overlap between GEs is not permitted, then if a GE is recognized in another GE they are split. Also for GEs a stop list to exclude non-relevant elements has been defined . As shown in the

Figure 2 the idea behind SAGG web page visual analysis algorithm is to support the SAGG's semantic search algorithm by providing as output a list of GO and for each one the related graph of GEs that will be analyzed from the SAGG's semantic search algorithm.

In this sense SAGG's approach follows the idea that web page segmentation can improve the semantic search process.

The contributions provided by SAGG's VIPS algorithm are the introduction of a further level of segmentation in VIPS approaches and the introduction of an approach that affirms that the segmentation it isn't only useful to identify different topics within a web page but can provide further information about the nature of each of these topics in the page.

In the next section the SSA algorithm a semantic search algorithm that exploits the SAGG's VIPS algorithm output to improve its semantic search process will be proposed.

2.2.2. Query Generator

This phase of the algorithm consists in analysing, for each GO, its associated triples list, in order to match triples in the linked data repository.
First of all, for each triple GE, we extract keywords from their textual content and try to match them over RDF nodes in the linked data repository.
The result of this step is a list of seeds for GEs that are bound through a relation. Using this seed the algorithm navigates into the linked data repository .
The next operation is to rank triples according to the strategy previously mentioned in the model description. Triples over a given threshold are analysed to extract a list of queries (in SPARQL) over the user dataset which are able to reproduce their content. The last operation of the algorithm consists in unifying these small queries and to create the best graph pattern which approximates the constraints to reproduce the content of the whole GO.
Further details about this algorithm called SSA algorithm will be provided in the next section.

2.2.2.1. SSA Algorithm

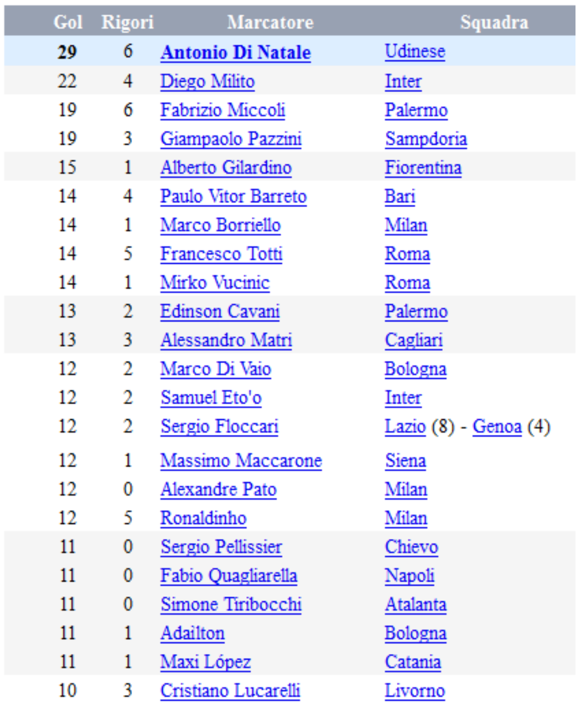
As mentioned above the retrieval of semantic web data is a challenging task, in the SAGG's approach it is needed to understand content of a example page to extract query exploited to populate generated GUIs.
In this paragraph the SSA algorithm has been presented, SSA is a semantic search algorithm implemented to accomplish the challenge of search semantic web data.
The SSA algorithm takes in input the output provided by the SAGG's VIPS algorithm (see 2.2.1.1) that is a set of GOs, composed by a set triples in the form of:

$$< GE_1;\ \text{relation}; GE_2 > \qquad (6)$$

The interaction between SAGG's SSA Algorithm and SAGG's VIPS Algorithm is shown in the
Figure **2**
The SSA algorithm for each of previous triples identifies a related list of triples composed as:

$$< \text{Keyword}_1;\ \text{relation}; \text{Keyword}_2 > \qquad (7)$$

To extract keywords from textual content of both GEs in the triple SSA uses the Chaos parser (Zanzotto, 2001). Chaos is a modular and lexicalized syntactic and semantic parser for Italian and for English.
In the SSA approach, the syntactic parsing features are used to identify into the textual content of the GEs the nouns , the verbs and the adjectives that are considered as possible candidate keywords (other words as example the conjunctions, the prepositions etc. are discarded).
The second phase of the SSA algorithm for each GO analyzes the list of previous triples to map theme in the RDF graphs by the following action: the SSA algorithm, tries to mapping each keyword into a node in a RDF graph.
The relation is investigated using an approach guided by the nature of the relation identified in SAG's VIPS algorithm (i.e. if the relation is a dependency relation,



Gol	Rigori	Marcatore	Squadra
29	6	**Antonio Di Natale**	Udinese
22	4	Diego Milito	Inter
19	6	Fabrizio Miccoli	Palermo
19	3	Giampaolo Pazzini	Sampdoria
15	1	Alberto Gilardino	Fiorentina
14	4	Paulo Vitor Barreto	Bari
14	1	Marco Borriello	Milan
14	5	Francesco Totti	Roma
14	1	Mirko Vucinic	Roma
13	2	Edinson Cavani	Palermo
13	3	Alessandro Matri	Cagliari
12	2	Marco Di Vaio	Bologna
12	2	Samuel Eto'o	Inter
12	2	Sergio Floccari	Lazio (8) - Genoa (4)
12	1	Massimo Maccarone	Siena
12	0	Alexandre Pato	Milan
12	5	Ronaldinho	Milan
11	0	Sergio Pellissier	Chievo
11	0	Fabio Quagliarella	Napoli
11	0	Simone Tiribocchi	Atalanta
11	1	Adailton	Bologna
11	1	Maxi López	Catania
10	3	Cristiano Lucarelli	Livorno

Figure 3 HTML Input Table

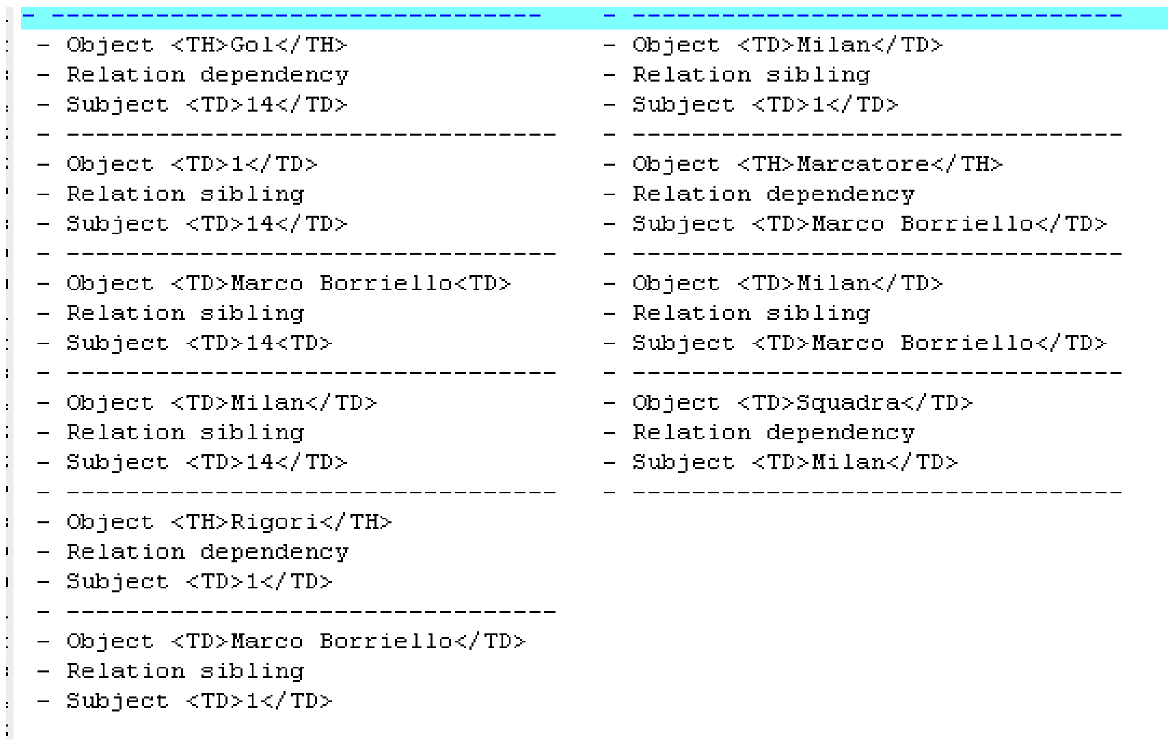
the SSA investigates first of all "*InstanceOf* "and "*SubClassOf* "and only if none of these are reify others relation in knowledge base are investigated).
After the mapping process all candidate RDF triples:

$$<\text{Node}_x\ ,\text{predicate},\text{Node}_y> \qquad (8)$$

which match

$$< GEx, \text{relation}, GEy >. \qquad (9)$$

are identified .
After the validation of triples, the validated triples are analyzed and composed to create a list of queries related to each GO.
The SSA algorithm differs from others semantic search approaches mentioned in (Dr.T.V.Rajinikanth, 2011) because they take as input only queries composed by list of keywords, SSA instead exploits the output of SAGG's VIPS Algorithm.
Our approach makes easier the composition of candidate matching graph patterns (lists of triples) instead of a simple list of keywords.
The SAGG system proposes a new interaction mode between the users and the semantic search systems, it asks users to indicate only a example of web page that contains both keywords and visualization patterns instead semantic or keywords query.
By using information provided by the input page analyzed by the SAGG's VIPS Algorithm, the SSA algorithm implements an algorithm of query expansion guided by the definition of relationship identified by the analysis provided by web page visual analysis step. The SSA algorithm tries to map not only the keywords, but all



```
- ---------------------------------    - ---------------------------------
 - Object <TH>Gol</TH>                 - Object <TD>Milan</TD>
 - Relation dependency                 - Relation sibling
 - Subject <TD>14</TD>                 - Subject <TD>1</TD>
 - ---------------------------------   - ---------------------------------
 - Object <TD>1</TD>                   - Object <TH>Marcatore</TH>
 - Relation sibling                    - Relation dependency
 - Subject <TD>14</TD>                 - Subject <TD>Marco Borriello</TD>
 - ---------------------------------   - ---------------------------------
 - Object <TD>Marco Borriello<TD>      - Object <TD>Milan</TD>
 - Relation sibling                    - Relation sibling
 - Subject <TD>14<TD>                  - Subject <TD>Marco Borriello</TD>
 - ---------------------------------   - ---------------------------------
 - Object <TD>Milan</TD>               - Object <TD>Squadra</TD>
 - Relation sibling                    - Relation dependency
 - Subject <TD>14</TD>                 - Subject <TD>Milan</TD>
 - ---------------------------------   - ---------------------------------
 - Object <TH>Rigori</TH>
 - Relation dependency
 - Subject <TD>1</TD>
 - ---------------------------------
 - Object <TD>Marco Borriello</TD>
 - Relation sibling
 - Subject <TD>1</TD>
```

Figure 4: Triple List

the triples, this allow to identify both resources and relations.

2.2.3. Configuration File Composer

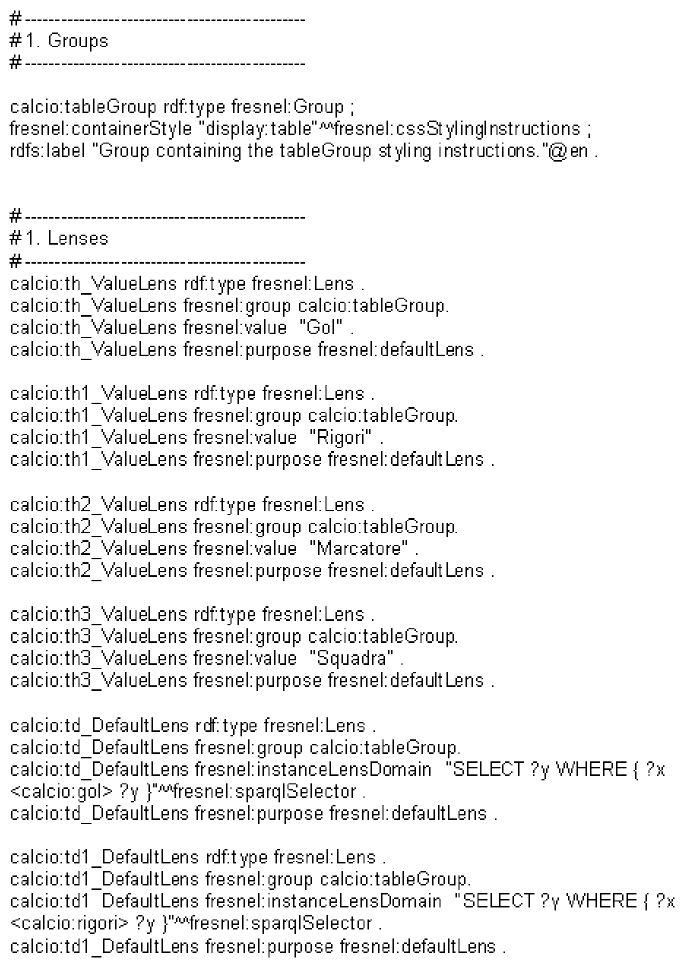
In (Pietriga, Bizer, Karger, & Lee, 2006), the authors present Fresnel, an RDF vocabulary for RDF information visualization. Fresnel's two basic concepts are *lenses* and *formats*. Lenses define which properties of one or more RDF resources to display and their order of presentation.
Formats determine how to render the resources, their properties and values.
The configuration file composer uses both the graphic pattern and the query generated by previous steps to create a configuration file following Fresnel vocabulary. The GO pattern are used to generate formats. The graphs provided by Query Generator are used to generate lenses.

2.2.4. Form Composer

The form composer steps automatically generate a form capable of showing the linked data organized according to the desired pattern of visualization. To perform this task we have implemented a Fresnel parser able to read a configuration file and create a form.

3. SAGG: Scenario

In this section an example of use of SAGG is presented . A web service able to take in input an html fragment and a URI of a knowledge base and to provide in output an html page is implemented .
As shown in the Figure 3 the input is composed by a part of a document HTML that contains a table.
The knowledge base of user is composed by a simple ontology that defines only two concepts "*Marcatore*"(football player) and "*Squadra*"(team).
The ontology defines also three relationship: an object property "*gioca in*"(play in) that relate the concept "*Marcatore*" with the concept "*Squadra*"; and two datatype property "*golSegnati*"and "*rigoriSegnati*"(they represent respectively goals and penalties scored by the player).
The Pattern Generator analyzes the HTML document given in input and recognizes the table as a GO. Then it identifies as GE all cells of the table (the row are discarded because there is no information beyond the cell in a row) and creates a list of triples.
Note that in this case a custom strategies is applied to analyze the table and in according to this strategy in the triples list are added only the relationship between the cell which are located in the same row or in the same column, besides the dependency relations between header and cell of a column are added. In the Figure 4 is shown a fragment of the triples list that contains all the triples generated for a row of analyzed table. The Query Generator takes in input previous triples list and analyzes this list to generate the list of atomic queries.
The Configuration File Composer module take in input both the lists of simple queries and values and the pattern provided by the Pattern Generator. The Configuration File Composer exploit these inputs to create the Fresnel Configuration File. A fragment of the created Configuration File is shown in the
Figure 5. At the end of the process the GUI composer module parses the configuration file and creates the HTML GUI that is shown in the Figure 6



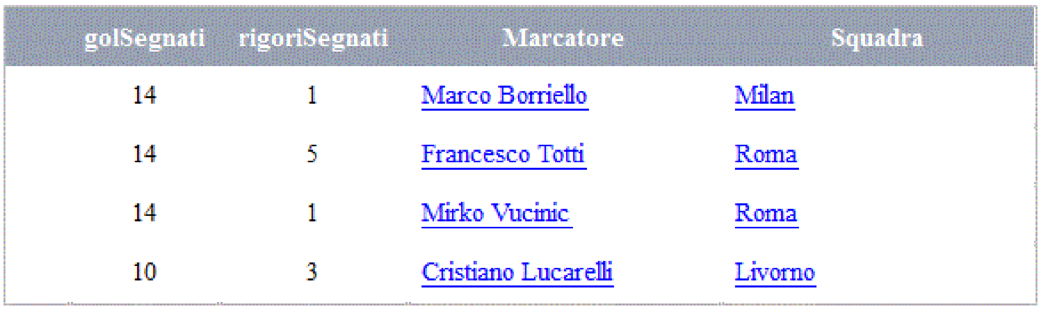
```
#----------------------------------------------
#1. Groups
#----------------------------------------------

calcio:tableGroup rdf:type fresnel:Group ;
fresnel:containerStyle "display:table"^^fresnel:cssStylingInstructions ;
rdfs:label "Group containing the tableGroup styling instructions."@en .


#----------------------------------------------
#1. Lenses
#----------------------------------------------
calcio:th_ValueLens rdf:type fresnel:Lens .
calcio:th_ValueLens fresnel:group calcio:tableGroup.
calcio:th_ValueLens fresnel:value  "Gol" .
calcio:th_ValueLens fresnel:purpose fresnel:defaultLens .

calcio:th1_ValueLens rdf:type fresnel:Lens .
calcio:th1_ValueLens fresnel:group calcio:tableGroup.
calcio:th1_ValueLens fresnel:value  "Rigori" .
calcio:th1_ValueLens fresnel:purpose fresnel:defaultLens .

calcio:th2_ValueLens rdf:type fresnel:Lens .
calcio:th2_ValueLens fresnel:group calcio:tableGroup.
calcio:th2_ValueLens fresnel:value  "Marcatore" .
calcio:th2_ValueLens fresnel:purpose fresnel:defaultLens .

calcio:th3_ValueLens rdf:type fresnel:Lens .
calcio:th3_ValueLens fresnel:group calcio:tableGroup.
calcio:th3_ValueLens fresnel:value  "Squadra" .
calcio:th3_ValueLens fresnel:purpose fresnel:defaultLens .

calcio:td_DefaultLens rdf:type fresnel:Lens .
calcio:td_DefaultLens fresnel:group calcio:tableGroup.
calcio:td_DefaultLens fresnel:instanceLensDomain  "SELECT ?y WHERE { ?x
<calcio:gol> ?y }"^^fresnel:sparqlSelector .
calcio:td_DefaultLens fresnel:purpose fresnel:defaultLens .

calcio:td1_DefaultLens rdf:type fresnel:Lens .
calcio:td1_DefaultLens fresnel:group calcio:tableGroup.
calcio:td1 DefaultLens fresnel:instanceLensDomain  "SELECT ?y WHERE { ?x
<calcio:rigori> ?y }"^^fresnel:sparqlSelector .
calcio:td1_DefaultLens fresnel:purpose fresnel:defaultLens .
```

Figure 5: Configuration File

As shown in the Figure 6 the GUI contains data retrieved from the knowledge base and it is formatted like the input page.
In conclusion the configuration file is saved in the knowledge base side by side with the domain ontology.

4. Conclusions

In this paper an innovative approach to generate visualization forms able to show linked data in an appealing manner has been presented.
The data model behind our system and the algorithms which realize it have been described.
The algorithms provided by SAGG cover the analysis of patterns of representation and the generation of structured queries exploiting the selected graphical objects.
The algorithms' implementation also cover the generation of Configuration Files according to Fresnel W3C specification and finally they realize the generation of customized GUI to show data.
Differently from others approach in literature, this approach is open to different domains because SAGG asks users to indicate their own knowledge base and generate the GUI exploiting the example web page.
The SAGG approach can be implemented and integrated in very different scenarios, as an extension for Semantic Enhanced Web Browsers, RDF Browsers, Ontology Editors and Annotation Tools.
In this paper, to achieve the aim of generate GUI in semi-automatic way, a novel VIPs algorithm the SAGG's VIPs algorithm has been proposed.
This algorithm introduces a further level of segmentation respect to the traditional VIPS algorithms, indeed not only the visual object (called GO in this approach) but also their content are analyzed to individuate the elements (called GE) that are part of the GO and theirs relations.
This approach allows to make assumptions about the content of each GO. The output of this algorithm is a list of graphs of GEs related to each GO.
Exploiting this output, also an innovative semantic search algorithm, the SSA algorithm, has been realized.
The SSA algorithm exploits the output of SAGG's VIPs algorithm to perform an optimized search process.
In addition to visualization, our approach can be exploited to facilitate the ontology population.
Indeed SAGG identifies not only the candidate queries but also values that could be used to fill the GEs; if these values are not in the knowledge base they can be in any case added to the generated GUI and they can be proposed to the users. If the users validate them they are stored in the knowledge base.
In conclusion we can affirm that the SAGG approach has



golSegnati	rigoriSegnati	Marcatore	Squadra
14	1	Marco Borriello	Milan
14	5	Francesco Totti	Roma
14	1	Mirko Vucinic	Roma
10	3	Cristiano Lucarelli	Livorno

Figure 6: Output HTML table

reached the goal of providing a RDF browser user-friendly and with a very tasteful graphical interface, furthermore the SAGG approach is compliant with the reusability and sharing principle of the Semantic Web vision because it provide a standard configuration file that can be reused and shared among users.
Moreover in the SAGG approach have been implemented two innovative algorithms (the SAGG's VIPS algorithm and the SSA algorithm).

5. Future Works

A future research direction for SAGG lies in the combination of several Fresnel files to generate more complex GUIs, possibly specifying interrelationships (i.e. semantic constraints) between them.
While this could simply be seen as a further refinement, we would stress the importance for the user of being able to specify compositional patterns for reusable atomic Fresnel units, in a sort of Semantic Mash-up.
This would open up the way to reusable, shareable libraries of active UIs (i.e., carrying the information on how to populate them from available data), which could be easily searched (according to different perspectives, what they show, how they show it, etc..), accessed, imported (into heterogeneous Semantic UI developing environment) and composed according to user/developer needs, in the spirit of the Semantic Web vision.
Another important future improvement is to include the formalization of relations between GOs to extend the VIPS algorithm and consequently to provide a more informative input to our semantic search algorithms.
The exploration of this kind of relations is based on the idea that – as affirmed above about the significance of the position of GEs in a GO – GOs' position in the example pages can be useful to understand their meaning, mostly in the cases where a GO is nested in another GO.
Moreover the SAGG's VIPS algorithm can be improved by the definition of further strategies that will improve the identification of the relations between GEs in several GOs.

6. References

Dadzie, A.-S., & Rowe, M. (2011). Approaches to Visualising Linked Data: A Survey . *Semantic Web Journal* , 89-124.

Dr.T.V.Rajinikanth, G. a. (2011). Intelligent Semantic Web Search Engines: A Brief Survey. *International journal of Web \& Semantic Technology (IJWesT)* (Vol.2, No.1).

Le Hégaret, P., Whitmer, R., & Wood, L. (2009, 06 01). *W3C Document Object Model.* Retrieved from World Wide Web Consortium (W3C): http://www.w3.org/DOM/

Lei, Y., Uren, V. S., & Motta, E. (2006). SemSearch: A Search Engine for the Semantic Web., (p. 238-245).

Mangold, C. (2007). Int. Journal Metadata Semantics and Ontology. A survey and classification of semantic search approches. *2* (1).

Myungjin Lee, W. K. (2010). Semantic Association-Based Search and Visualization Method on the Semantic Web Portal. *International journal of Computer Networks & Communications* , *2* (1), 140-152.

Pazienza, M. T., Scarpato, N., & Stellato, A. (2010). Semi-automatic Generation of GUIs for RDF Browsing. *14th International Conference on Information Visualisation, IV*, (p. 267-272). London.

Pietriga, E., Bizer, C., Karger, D., & Lee, a. R. (2006). Fresnel - A Browser-Independent Presentation Vocabulary for RDF. *5th International Semantic Web Conference*. Athens, GA, USA.

Thanh Tran, H. W. (2009). Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. *ICDE 2009*, (p. 405-416). Shanghai.

Tran, T., Cimiano, P., Rudolph, S., & Studer, R. (2007). Ontology-Based Interpretation of Keywords for Semantic Search. *Lecture Notes in Computer Science* , *4825*, 523-+.

Wu, C. a. (2006). A Web page Segmentation Algorithm for extracting product information. *Proceedings of IEEE International Conference on Publication*, 1374--1379.

Zanzotto, R. B. (2001). Flexible Parsing Architectures for NLP Applications. *AI*IA*, 308-313.