KNOWLEDGE-BASED VISUALIZATION SYSTEMS

Noemi Scarpato

Supervisor:
Prof. Maria Teresa Pazienza

Rome, June 2012
## Contents

### 1 Thesis Introduction

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Information and Knowledge Visualization</td>
<td>8</td>
</tr>
<tr>
<td>1.2 Visualization Process</td>
<td>10</td>
</tr>
<tr>
<td>1.3 Knowledge-Based Visualization System</td>
<td>11</td>
</tr>
<tr>
<td>1.3.1 Existing Approaches</td>
<td>12</td>
</tr>
<tr>
<td>1.3.2 Problems and open challenges in Knowledge-Based Visualization Systems</td>
<td>12</td>
</tr>
<tr>
<td>1.4 Thesis Contributions</td>
<td>13</td>
</tr>
<tr>
<td>1.5 Thesis Outlines</td>
<td>15</td>
</tr>
</tbody>
</table>

### 2 State Of The Art

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Information and Knowledge Visualization</td>
<td>20</td>
</tr>
<tr>
<td>2.1.1 Information Visualization</td>
<td>20</td>
</tr>
<tr>
<td>2.1.2 Knowledge Visualization</td>
<td>21</td>
</tr>
<tr>
<td>2.1.3 Difference Between Information Visualization and Knowledge Visualization</td>
<td>22</td>
</tr>
<tr>
<td>2.1.4 Synergies between Information Visualization and Knowledge Visualization</td>
<td>23</td>
</tr>
<tr>
<td>2.2 Approaches to visualizing metadata</td>
<td>23</td>
</tr>
<tr>
<td>2.3 Fresnel</td>
<td>27</td>
</tr>
<tr>
<td>2.3.1 Lens</td>
<td>29</td>
</tr>
<tr>
<td>2.3.2 Format</td>
<td>31</td>
</tr>
<tr>
<td>2.3.3 Fresnel: purposes and uses</td>
<td>33</td>
</tr>
<tr>
<td>2.4 Knowledge-Based Visualization Systems</td>
<td>34</td>
</tr>
<tr>
<td>2.4.1 State of the art of RDF browsers</td>
<td>36</td>
</tr>
<tr>
<td>2.4.1.1 Fresnel Based RDF Browsers</td>
<td>39</td>
</tr>
<tr>
<td>2.4.2 Problems and open challenges in Knowledge-Based Visualization Systems</td>
<td>40</td>
</tr>
<tr>
<td>2.5 Conclusions</td>
<td>41</td>
</tr>
</tbody>
</table>
CONTENTS

3 Semantic Search 43
   3.1 Semantic Search Approaches ....................................... 43
   3.2 Query Formulation .................................................. 44
      3.2.1 SPARQL Query Language ....................................... 46
   3.3 Semantic Search in RDF Browser .................................. 47
      3.3.1 Manual Query Composition ................................... 48
      3.3.2 Semantic Query Expansion ................................... 49
      3.3.3 Visual Query Composition ................................... 51
   3.4 Conclusions .......................................................... 53

4 Web page visual analysis 55
   4.1 Web page segmentation approaches ................................. 55
      4.1.1 DOM based segmentation Algorithms .......................... 56
         4.1.1.1 The Document Object Model ............................... 56
      4.1.2 Template Detection algorithms ................................ 56
      4.1.3 Web page visual analysis Approaches ........................ 57
         4.1.3.1 VIPS algorithm ............................................ 58
         4.1.3.2 Other Web page visual analysis approaches ............ 60
   4.2 Using Web Page Analysis to improve Search Engine Performance 61
   4.3 Conclusions .......................................................... 63

III Thesis Contributions 65

5 Semi-Automatic Gui Generator (SAGG)  67
   5.1 Overall Concept ...................................................... 67
   5.2 Scenario ............................................................ 70
   5.3 Model ................................................................. 71
   5.4 User Interaction ..................................................... 75
   5.5 SAGG Web page visual analysis algorithm ........................ 76
   5.6 SAGG’S Semantic Search Algorithm ................................ 79
   5.7 Architecture .......................................................... 81
      5.7.1 Pattern Generator .............................................. 85
         5.7.1.1 Strategies .................................................... 87
      5.7.2 Query Generator ................................................ 88
      5.7.3 Configuration File Composer ................................. 90
      5.7.4 GUI Composer .................................................. 93
   5.8 SAGG at work .......................................................... 93
   5.9 Conclusion ............................................................ 98
## CONTENTS

6  SAGG in Semantic Turkey  \hfill  101
   6.1  Semantic Turkey \hfill  101
6.2  Semantic Turkey Extensions \hfill  104
6.3  Semantic Turkey ontologies visualization feature \hfill  106
6.4  ST-SAGG \hfill  108
   6.4.1  User Interaction \hfill  111
6.5  Conclusions \hfill  113

IV  Conclusions  \hfill  115

7  Conclusions and future Works  \hfill  117
   7.1  Future Works \hfill  119
List of Tables

5.1 Rank Values ........................................... 75
## List of Figures

1.1 RDF triple .................................................. 6  
2.1 Semantic Web layer cake .................................. 24  
2.2 The Linking Open Data cloud diagram .................. 25  
2.3 Interactive tree in Semantic Turkey .................... 26  
2.4 Fresnel ...................................................... 28  
2.5 OWLViz graph view of a small Ontology ................ 35  
2.6 OWLViz graph view of a big Ontology .................. 35  
2.7 Marbles Formatted RDF display view .................... 36  
2.8 Domain-specific visualization ............................ 37  
2.9 Widgets-based .............................................. 37  
3.1 Semantic Turkey SPARQL Panel .......................... 48  
3.2 Tabulator Visual Query Interface ........................ 52  
3.3 Tabulator Visual Query Interface ........................ 53  
5.1 System Design .............................................. 70  
5.2 Activity diagram for SAGG ............................... 74  
5.3 Interaction between SAGG’s SSA Algorithm and SAGG’s VIPS Algorithm ................................................. 79  
5.4 Semi-Automatic GUI Generator Architecture .......... 82  
5.5 HTML input Table ......................................... 95  
5.6 Triple List .................................................. 96  
5.7 Queries and values List ................................... 96  
5.8 Configuration File ......................................... 97  
5.9 Generated GUI ............................................. 98  
6.1 Semantic Turkey ........................................... 102  
6.2 Semantic Turkey Graph View ............................. 107  
6.3 Architecture of ST-SAGG .................................. 110  
6.4 Activity diagram for ST-SAGG ............................ 112
<table>
<thead>
<tr>
<th>No.</th>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fresnel Group</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>Simple Selector Fresnel Lens</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Fresnel Selector Lens</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>SPARQL Selector Lens</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>Fresnel Format</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>Explicit Link</td>
<td>33</td>
</tr>
<tr>
<td>7</td>
<td>GO identification Algorithm</td>
<td>86</td>
</tr>
<tr>
<td>8</td>
<td>GE identification Algorithm</td>
<td>87</td>
</tr>
<tr>
<td>9</td>
<td>GE identification Algorithm</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>Query Generator Algorithm</td>
<td>90</td>
</tr>
<tr>
<td>11</td>
<td>Query Generator Algorithm 2</td>
<td>91</td>
</tr>
<tr>
<td>12</td>
<td>Creation of Lenses Algorithm</td>
<td>92</td>
</tr>
<tr>
<td>13</td>
<td>Creation of GUI Algorithm</td>
<td>94</td>
</tr>
</tbody>
</table>
Part I

Thesis Introduction
The Semantic Web [Tim Berners Lee et al.(2001)], also defined as “Web of data”, implies that data in the Web should be not only machine-readable but also machine-understandable.

Starting from this assumption, it is necessary to introduce the concept of metadata, that can be defined as ”data that describe data”: this kind of information allows communication between computers and humans. The metadata can make the meaning of data on the Web explicit so as to provide computers with enough information to handle such data.

The key idea of this vision is to make the Web machine-understandable, differently from traditional Web where data are only machine-readable.

To define metadata, it is needed to refer to a shared model of the world, then the definition of ontologies has been introduced to provide this model in the Semantic Web. The ontologies establish a common conceptual description and a joint terminology between members of communities of interest (human or autonomous software agents). While Ontology term is borrowed from Philosophy, several definition of the concept of ontology have been provided in the past.

In this thesis we take the following definition for ontology concept, this definition is usually adopted in computer science:
“an ontology is an explicit, formal specification of a conceptualization needed to organize knowledge in specific domains and applications.” \[Thomas Gruber(1993)\] .

Making data machine-understandable opens new scenarios on the Web, where humans and machine can communicate without misunderstandings. This scenario presents several challenges in many fields of research.

Different standards have been defined to implement the Semantic Web vision:

- **RDF**\[Ben Adida and Mark Birbeck(2008)] RDF (Resource Description Framework) is a language for representing information related to resources in the Semantic Web. RDF is a direct labeled graph that it is defined to represent metadata. RDF also provides an XML-based syntax (called RDF/XML) for writing and exchanging these metadata. RDF is based on the concept of triple. An RDF triple is composed by two resources and a property as below;

![RDF triple](image)

Figure 1.1: RDF triple

- **RDFS**\[is a general-purpose language for defining the properties of the resource

\[1http://www.w3.org/TR/2000/CR-RDF-schema-20000327/\]
and the kind of resources being described. RDFs is an extension of RDF;

- **OWL [McG(2004)]** is a language for defining Web ontologies, it is developed as a vocabulary extension of RDF and it is derived from the DAML+OIL Web Ontology Language;

- **SPARQL [Ben Adida and Mark Birbeck(2008)]** is a query language for RDF;

- **Fresnel [Emmanuel Pietriga et al.(2006)]** is a visual language for RDF(further details in [2,3]).

One of the most active areas of the Semantic Web is the development of a variety of tools (e.g. browser, search engine, semantic annotation tools etc...) for the authoring, extraction, visualization, and inference of metadata.

For what concerning the Semantic Web, visualizing the structure of a formally defined information resource domain is important not only for designers and developers, but also for the business and individual users of the first generation of the Web. Visual representation of metadata in their raw form, is difficult to understand for common users, (see figure [1.1]), then in the last years a new field of research has been investigated to implement the information visualization on the Semantic Web.

In the Semantic Web, visualization is a crucial aspect to provide users with GUIs that show metadata in a visually comprehensible way.

The objective of this thesis is to achieve this challenge, to do that, it is due to analyze the state of the art of visualization research field, proposing a novel solution to carry out a system able to create GUIs in a semi-automatic way.
Chapter 1. Introduction

1.1 Information and Knowledge Visualization

Information Visualisation can be defined\(^2\) as:

“...a process of transforming information into a visual form enabling the viewer to observe, browse, make sense and understand the information. It typically employs computers to process the information and computer screens to view it using methods of interactive graphics, imaging and visual design. It relies on the visual system to perceive and process the information.”

Information visualization (InfoVis from now on) has been recently devoted to exploit metadata to provide proper representation for data in the Semantic Web. Once the metadata are available, it is relatively straightforward to use them, in order to visualize them.

The knowledge-based visualization (KVis from now on) research field is involved in implementation of systems that exploit domain knowledge to define proper patterns of visualization. In this sense, RDF visualization can surely benefit from the results and findings in the InfoVis field. However, most existing knowledge-based visualization applications work only on specific domains/tasks, reflecting the difficulties in generalizing and reapplying visualization approaches to new problems or domains.

In the Semantic Web, the task of visualization of metadata consists of visualizing the RDF data in a visually comprehensible form. The objective of this task is to make RDF data human-understandable. To accomplish the task of RDF visualization it is due to specify which information contained in an RDF graph and how this information should

\(^2\)http://www.infovis.org/
1.1. Information and Knowledge Visualization

be presented.

There are two major approaches to Semantic Web data visualization: adopting and applying existing InfoVis solutions or developing completely new techniques specifically tailored for the knowledge representation paradigms of the Semantic Web.

The information that is originated from RDF data can be interpreted as a subgraph. This subgraph, can be extracted by using queries: selecting and composing these queries (for which a standard is available from early 2008, in the form of the SPARQL query language) requires a combination of domain/technical expertise to be applied.

Another key point of Semantic Web Data Visualizations is the definition of the graphic elements that are associated to domain information. These can be selected to expose a representation on their own. The Semantic extensions to RDF, such as RDFS/OWL or other standards such as SKOS may play a pivotal role in representing this information, through “ad hoc” ontologies of visualization, these ontologies refer to graphical aspects such as templates decorating UI widgets, geometrical aspects (size, width, depth etc), or other aspects strictly related to the source data (the order in which certain collections of resource are displayed, or the way they are clustered etc).

It is very important to point out that the pairs of type:

\[ < RDF - resource, representation > \]  \quad (1.1)

are relevant “per se”, which can be exposed, collected and reused by generic RDF browsers and viewers according to the same paradigm which is proposed by the Semantic Web for resource shareability and reusability.
Chapter 1. Introduction

1.2 Visualization Process

The visualization of data can be defined as a process that aims to transforming information into a visual form. This can be achieved on the one hand through the selection of proper sub-graph in an RDF graph, that should match user needs and on the other hand through the definition of how this sub-graph could be presented. In [Rutledge et al. (2005)] the authors state that this visualization process can be divided in three steps:

- **Selection**: aims to identify sub-graph.
- **Structuring**: aims to define proper visualization structures.
- **Formatting**: aims to define formats related to above mentioned structures to styling visualization pattern (e.g. colors, fonts, background).

Each of this steps can be completed by the application of several techniques.

Selection step can be provided by the following approaches:

- **Manual filtering**: to select the subgraph, users select the resources manually.
- **Incremental navigation**: to select subgraph, users navigate trough RDF graph.
- **Querying the metadata repository**: to select subgraph, users submit a query.

As well as on the Web, users submit query to retrieve information, in the Semantic Web users should be able to submit query to select relevant data.

Semantic Search is the field of research that defines methods to search data in the Semantic Web (see chapter 3). Also Semantic Search approaches can be improves through the integration of techniques of Web page visual analysis (see chapter 4).
Structuring & Formatting steps deals with the definition of visualization structures and theirs styles. In [Matthias Palmer et al. (2007)], the authors discuss the design of the Annotation Profile Model, which consists of a data-capturing (the Graph Pattern Model) and a presentation part (the Form Template model). Furthermore in [Emmanuel Pietriga et al. (2006)] the authors defines an RDF display vocabulary: Fresnel. Fresnel allow to define RDF visualization information that can be linked to an RDF resources to define representation information of data.

Fresnel defines two basic concepts:

- 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.

By using Fresnel it is possible to complete Structuring & Formatting steps of visual process.

The basic idea of these approaches is the definition of a evident separation between the data and their representation. These approaches enable developers to define a chain of operations that allow the realization of knowledge-based visualization systems, that are able to create a user-friendly GUI to browse the Semantic Web.

1.3 Knowledge-Based Visualization System

Key idea beyond Knowledge-based visualization systems is to exploit metadata to visualize them. The definition of the user interface is yet an open challenge in the Semantic Web, one of the main difficulties for the research community is to make linked open data in the Semantic Web. Linked open Data have by their nature an high coupling
Chapter 1. Introduction

value, then when we visualize linked open data we need to show complex and very large graphs of resources. Retrieval and organizations of metadata in the Semantic Web is an additional challenge in the visualization of Semantic Web data understandable. Recently several approaches have been developed to create representation forms of metadata.

1.3.1 Existing Approaches

Existing knowledge-based visualization approaches are:

- Graph-based: provides a graph view representation;
- Faceted browsing: provides a forms view filled with formatted RDF representation of resources;
- Domain-specific: provides an interfaces tailored on a specific resource;
- Widgets-based: provides a rich graphical representation of resources by using graphical metaphor (e.g. table, list etc. in XHTML, XUL, SVG etc) and stylesheet.

1.3.2 Problems and open challenges in Knowledge-Based Visualization Systems

Problems on current knowledge-based visualization approaches are:

- Graph-based: scalability;
- Faceted browsing: lacks the possibility to show clearly the relationship;
- Domain-specific: no reusability;
1.4. Thesis Contributions

- Widgets-based: while this can be considered a good solution in term of representation quality, the existing implementations of widgets-based approaches, require manual submission of a Configuration Files.

Each of the previous approaches presents some problems: on the one hand some of these are not appealing for users, and on the other hand some others require the definition of a Configuration Files. One of main limits to a satisfactory implementation of knowledge-based systems is the lack of automatization in the process of visualization. The automatization is a crucial aspect that allow all kind of users (expert and mostly common users) to navigate the open linked data.

1.4 Thesis Contributions

The aim of this thesis is to respond to the challenge of the automatization of the linked open data visualization process. To reach this objective, firstly we will analyze the issue of visualization and in particular the linked open data visualization approaches. Then we will provide a critical analysis of these approaches and a novel solution to solved open challenges.

We propose a new approach to generate GUIs in semi-automatic way for RDF Browser [Maria Teresa Pazienza and Noemi Scarpato(2010)], also we provide the implementation of this approach in the SAGG system. To the best of our knowledge, our approach is new in knowledge-based visualization research field.

The overall design of the SAGG system requires users to indicate only examples of representation from which the system extracts automatically a GUI representation for the data contained in the knowledge base of the user itself. The SAGG system generates a Fresnel Configuration File, that can be reused and shared among different users.
Chapter 1. Introduction

SAGG is a knowledge-based visualization system that exploits some examples submitted from users to automate the visualization process.

The following components of SAGG system have been defined to implement the visualization process:

- **SAGG VIPS algorithm** (see 5.5): we propose a new approach to Web Page visual segmentation (VIPS) algorithms. That is implementing by designing an algorithm that introduces a further level of segmentation of the Web page, in comparison with the traditional approach.

The algorithm identifies relations between recognized visual objects through the definition of a model of visual element. The model exploits the visual objects nature and their relations.

Finally some *special* visual objects have been identified and custom strategies able to recognize both visual object and their relations.

Output of the SAGG VIPS algorithm is a graph of visual objects and their relations. This output will be taken in input from SAGG Semantic Search Algorithm. ([Maria Teresa Pazienza and Noemi Scarpato(2012)](Maria Teresa Pazienza and Noemi Scarpato(2012))

- **SAGG Semantic Search Algorithm** (see 5.6): we propose a novel Semantic Search algorithm. This algorithm performs a query expansion by the analysis of both the graph of visual objects and the relations provided by VIPS algorithm. The SAGG Semantic Search algorithm firstly extracts keywords from visual object content and then produces a list of triples. Finally the algorithm maps the triples into the knowledge base, this mapping process allows to extract the queries able to fill visual objects detected in VIPS algorithm with the resources contained in the knowledge base. ([Maria Teresa Pazienza and Noemi Scarpato(2012)](Maria Teresa Pazienza and Noemi Scarpato(2012))))
1.5 Thesis Outlines

- SAGG Structuring & Formatting Fresnel-based approach (see 5.7.3): we define a Configuration File composer module (see 5.7.3) to create Groups, Lenses and Formats according to the Fresnel standard language.

Finally we describe the integration of SAGG into Semantic Turkey [Maria Teresa Pazienza et al. (2011)] to allow users to benefit both of the Semantic Turkey peculiarities and also of the SAGG visualization features. The integration between SAGG and Semantic Turkey is implemented by the creation of ST-SAGG, an extension of Semantic Turkey System. The integration of SAGG and Semantic Turkey improves both Semantic Turkey and SAGG systems: on one hand users of Semantic Turkey benefit from both the visualizations peculiarities of SAGG and the creation of the Configuration File for their own RDF resources. On the other hand, users of SAGG take advantage of Semantic Turkey ontology editors features to store, modify and share visualization information.

1.5 Thesis Outlines

Chapter 2 provides a survey of the main methods, nowadays adopted, in Information Visualization and Knowledge Visualization. In particular, it is proposed a review of the state of the art and it is underlined the limits of these approaches.

Chapter 3 gives a survey of the main strategies and approaches adopted on Semantic Search has been provided. In particular, it analyzes the most recently approaches and algorithms used to retrieve Semantic Web data.

Chapter 4 proposes a survey of Web page visual analysis approaches. In particular it analyzes the advantage to use Web page visual analysis to support Semantic Search.

Chapter 5 presents the SAGG [Maria Teresa Pazienza and Noemi Scarpato (2010)]. SAGG is a system to Semi-Automatic GUI Generation able to browse Semantic Web
Chapter 1. Introduction

data. Chapter 5 provides also a comparison between these algorithms and the SAGG Semantic Search Algorithm (SSA).

Chapter 6 introduce the ST-SAGG. ST-SAGG is an extension of Semantic Turkey [Maria Teresa Pazienza et al. (2011)] able to integrate SAGG Knowledge visualization feature into Semantic Turkey.

Finally, chapter 7 draws the conclusions and outlines future research directions.
Part II

State Of The Art
Visualization of Linked Data (e.g., RDF data) is a key issue of Semantic Web. While machine can read linked data, these data are obscure for people if seen in their raw representation. Therefore, the development of visualization techniques to visualize linked data is becoming a key research topic.

This chapter reports on the general issue of data visualization. Section 2.1 describes the characteristics, differences, and possible synergies between two different topics in data visualization: information visualization and knowledge visualization. Section 2.2 discusses the existing approaches to visualize linked data and introduces to key requirements that visualization systems must fulfill. Section 2.3 describes the Fresnel presentation vocabulary for RDF [Emmanuel Pietriga et al. (2006)]. Section 2.4 describes Knowledge-Based Visualization Systems, introduces the state of the art of RDF Browser (Sec. 2.4.1) and gives details on the Fresnel-based RDF Browsers (Sec. 2.4.1.1), and, finally reports on the troubles and challenges related to existing approaches of RDF browsers (Sec. 2.4.2). Finally, section 2.5 draws conclusions about the knowledge visualization approaches and systems.
Chapter 2. Methods for Information Visualization and Knowledge Visualization

2.1 Information and Knowledge Visualization

In visualization research field it is possible to identify two distinct research areas: Information Visualization(InfoVis) and Knowledge Visualization(KVis)( see later sections for details ). The idea behind all visualization methods is that cognitive processing of complex subject matter, may be enhanced if graphical representations are able to make explicit structures behind knowledge and information. Researchers in the fields of Information Visualization and Knowledge Visualization are trying to develop and use tools for fostering access to information and knowledge resources. While these fields of research have grown up separately, many design methods for one can be applied to the other. Recently researchers in these fields have investigated, to achieve the goal to use both InfoVis and KVis methods to improve the understanding of visualized data.

2.1.1 Information Visualization

In literature different definitions of Information Visualization have been proposed:

“Information Visualization aims to produce graphical representations of abstract information structure for human users ”[Vladimir Geroimenko and Chaomei Chen(2006)].

“Information visualization (InfoVis) is the communication of abstract data through the use of interactive visual interfaces.”[Keim et al.(2006)].

“Information visualization is a set of technologies that use visual computing to amplify human cognition with abstract information.”[Stuart Card(2007)].

In all of previous definitions authors affirms that the visualization aims to provide graphical representation for human users. Then, in information visualization approaches,
most efforts are about the realization of visualization methods able to amplify cognitive performance of users in the interaction with computer. In general Information Visualization can be defined as a process that selects, organizes and shows information by using some patterns of visualization. This process has the purpose of make easy the consuming of information, hence the Information Visualization systems, generally do not provide functionality to allow users to edit represented data.

2.1.2 Knowledge Visualization

With respect to the Information Visualization research field, Knowledge Visualization is a younger research field. In Bertschi et al. (2011) authors provide the state of the art of the Knowledge Visualization and try to collect and organize the several definitions that were proposed in last years. In this paper are collected several ideas about the Knowledge Visualization definition provided by different researchers. In this paper Andrew Vande Moere affirms that since information becomes knowledge when it is possible to share context and meaning, the knowledge visualization can be defined as an Information Visualization in a shared context.

In general Knowledge Visualization can be defined as a process able to selects, organizes and shows knowledge with the aim to support sharing of knowledge between at least two human users.

Another key aspect of knowledge Visualization, is to provide instruments to enrich knowledge allowing modification of represented resources from human users that works in a Group.

Knowledge Visualization takes advantage from many different fields of research, such as Information Visualization and Knowledge Management.
2.1.3 Difference Between Information Visualization and Knowledge Visualization

One of the main differences between Information Visualization and Knowledge visualization refers to the objective of these research fields. While Information Visualization purpose is to implement visual applications to support comprehension of large amounts of data (often numerical data), instead the objective of the Knowledge Visualization is to enhance the transfer of knowledge among people and to improve the creation of knowledge in groups, using one or more visual representations.

A further difference between these two approaches relates to their goal. While Information Visualization aim is to show in a charming manner all considered data, instead Knowledge Visualization objective is the definition of subset of relevant data that should be showed to expedite knowledge consuming and enrichment.

Another difference between Information Visualization and Knowledge Visualization relates to nature of represented data: while the Information Visualization deals with visualization of explicit data (facts, numbers etc.), the Knowledge Visualization instead deals with visualization of metadata. This means that the Knowledge Visualization needs to make an extra effort to visualize the relationships between resources. While the Information Visualization approach, needs to show only considered data in the best possible way, the Knowledge Visualization approach, instead, concerns the use of graphical representation to highlight relationships between resources and for supporting the creation of new knowledge.
2.2. Approaches to visualizing metadata

2.1.4 Synergies between Information Visualization and Knowledge Visualization

The Information Visualization and The Knowledge Visualization research fields have been developed separately, even though they may benefit one from the other. The idea behind both of this visualization researches is that both the management and the comprehension of information and knowledge can be enhanced by using a proper pattern of visualization. Therefore synergy between these research fields is worthwhile. For instance, the Knowledge Visualization can improves its performance through the integration of the multi-dimensional representations of the knowledge, that was studied by several research projects in the field of the Information Visualization. The Information Visualization could benefit from the exploitation of the knowledge, that can be considered as an additional set of information that should be represented in the Information Visualization approaches. This provides many advantages in Information Visualization leading to the representation of additional information that are useful to clarify meaning of the data.

2.2 Approaches to visualizing metadata

The upper layer of the Semantic Web layer cake deals with the definition of the User Interface (see figure 2.1). While other layers of the Semantic Web layer cake have been strongly inspected and many solutions have been provided to solve the challenges offered by them, definition of the user interface is yet an open challenge. One of the main challenge to the research community is to make sense of linked open data present in the Web. Different from the traditional data, that have a less coupling value, the linked open data have by their nature high coupling value; this means that
when we want to visualize the linked open data we need to show a complex and very large graph of resources instead a simple list of data.

Since to the metadata are available it is possible to exploit them, to carry out a proper representation of metadata themselves.

Since the definition of the Semantic Web [Tim Berners Lee et al. (2001)] the amount of the linked open data stored in the Web was growing exponentially. By visiting the site: http://thedatalink.org/ it is possible to understand the current amount of the linked open data. Figure 2.2 shows the Linking Open Data cloud.

The retrieval of metadata in the Semantic Web and the organizations of results is an additional challenge in the visualization of Semantic Web data. As described in

---

2.2. Approaches to visualizing metadata

Figure 2.2: The Linking Open Data cloud diagram.

previous sections while in the Information Visualization research field the objective is to represent all data, in the Knowledge Visualization research field the objective is to identify a subset of relevant data and to show them in an appealing manner. In Knowledge Visualization one of the main issue is the implementation of methods to retrieve and to organize metadata in a proper visual form.

The visualization of metadata is almost young research field, in spite of this the formalization of the visualization process was carried out.

The visualization process of metadata can be divided in three steps: selection, structuring and a formatting step ([Rutledge et al. (2005)]).

Selection can be provided by using: manual filtering, incremental navigation or by querying the metadata repository.

Manual Filtering asks users to indicate what are the resources, they want to visualize. Incremental Navigation is realized through the realization of structured views of RDF
as example navigable trees or lists, this kind of views allows users to filter data just selecting a node of the tree. For example in the figure 2.3 it’s shown the interactive tree of Semantic Turkey [Maria Teresa Pazienza et al. (2011)].

Quering metadata approach is involved in interrogate a knowledge base with the aim to select a set of metadata that is relevant for the users. In chapter 3 we provide further information regarding the existing approaches to querying Semantic Web data.

Structuring and formatting steps deals with the need to define structure and its related formats. In the [Emmanuel Pietriga et al. (2006)] authors defines an RDF display vocabulary for RDF information visualization. Fresnel basic concepts are Lenses and Formats that allows to define respectively structure (Lenses) and formatting (Formats) to represent metadata (for detailed description of Fresnel see section 2.3).

A relevant contribute in the formalization of metadata visualization process, is
2.3. Fresnel

the definition of the Annotation Profile [Matthias Palmer et al. (2007)]. The Annotation Profile is an approach that allows to identify the clear separation between data and their representation, it is a model that identifies the data that must be displayed and its template of representation. The concept of Annotation Profile is derived from the concept of the Application Profile. While the Application Profile specifies what are the metadata to use in a specific application, the Annotation Profile is aimed to allow the automatic generation of the user interfaces for the metadata.

The Annotation Profile is composed by a graph pattern model and a form template model. The graph pattern model is responsible for capturing and creating subgraphs of the RDF graphs, the form template model instead defines the representations, the ordering and the grouping of each of these subgraphs.

One of the main shortcomings of the Annotation Profile approach is that requires metadata and/or domain experts that are able to define the Annotation Profiles according to this metadata.

General approaches to visualize metadata, may be applied both to metadata stored in a local repository and in the metadata presents in the Web. They provide a formal description of the visualization process.

We provide a detailed description of the approaches to visualize metadata in the Semantic Web in section 2.4, and an analysis of the problems and of the open challenges related to these approaches in section 2.4.2.

2.3 Fresnel

Fresnel [Emmanuel Pietriga et al. (2006)] is an RDF display vocabulary for RDF information visualization, it is supported by the W3C [Chris Bizer (2005)] and was released
Chapter 2. Methods for Information Visualization and Knowledge Visualization

in 2005. According to the Semantic Web’s standards, Fresnel was developed as an OWL [McG(2004)] ontology.

The basic idea behind Fresnel is the complete separation between the selection of data to be showed and their visual representation. As mentioned above, this idea is the basis behind the formalization of the visualization process. Fresnel provides the instruments to carry out this separation through the definition of Lenses and Formats. Therefore adopting this vocabulary as standard language for representing visual information seems to be a worthwhile solution.

Fresnel is able to define how styling RDF by using existing style languages such as CSS [css(1994)] that can be introduced into Fresnel Formats definitions.

Fresnel defines two main concepts: Lenses and Formats, we show the components of Fresnel and theirs relations in the figure 2.4.

Figure 2.4: Fresnel

Lenses and Formats are usually grouped together using a Fresnel Group. A Group
2.3. Fresnel

defines styling information for the container elements, it is possible to associate to a set of Lenses and Formats the same Group by using the *Fresnel:group* property. In the algorithm 1 we show an example of a Group.

Algorithm 1 Fresnel Group

```
ontExample:tableGroup RDF:type Fresnel:Group ;
    Fresnel:containerStyle “display:table” ;
    Fresnel:cssStylingInstructions ;
    rdfs:label “Group containing the tableGroup styling instructions.” en .
```

2.3.1 Lens

The Lenses define what are the properties related to one or more RDF resources that could be displayed and their order of presentation. To determine the sub-graph of resources that could be represented it is necessary to introduce the *Lens domain* in the Lens. Fresnel defines two kind of properties to determine the Lens domain:

- *Fresnel:instanceLensDomain*

- *Fresnel:classLensDomain*

While the value of *Fresnel:classLensDomain* represent an RDF class on which apply the Lens (i.e. the Lens is applied on each instance of the class), the value of Lens *Fresnel:instanceLensDomain* is a selector that allow to select a set of RDF instances on which apply the Lens. There are three kind of selector:

- *Simple Selector based on RDF URI references*: indicates the classes by using explicit RDF URI

- *FSL (Fresnel Selector Language) selector*: selects the instances by using the FSL [Emmanuel Pietriga(2005)] language.
Chapter 2. Methods for Information Visualization and Knowledge Visualization

- **SPARQL Selector**: selects instances by using the SPARQL [Eric Prud’hommeaux and Andy Seaborne(2008)] query language.

For instance: if we have an ontology whose prefix is `ontExample`, and in this ontology there is a concept named `Person` a simple selector Lens for this concept it is shown in the algorithm 2.

**Algorithm 2 Simple Selector Fresnel Lens**

```plaintext
:ontExamplePersonLens RDF:type Fresnel:Lens ;
    Fresnel:classLensDomain ontExample:Person .
```

An example of Fresnel Selector is provided into algorithm 3.

**Algorithm 3 Fresnel Selector Lens**

```plaintext
:ontExamplePersonLens RDF:type Fresnel:Lens ;
    Fresnel:instanceLensDomain "ontExample:Person[ontExample:knowsfoaf:Person]" 
    "Fresnel:fsselector.
```

The Last kind of Fresnel selectors is the SPARQL selector. This kind of selector provides the possibilities to integrate SPARQL queries into Fresnel Lenses. An example of a Lens that use a SPARQL selector is showed in the algorithm 4.

**Algorithm 4 SPARQL Selector Lens**

```plaintext
:ontExamplePersonLens RDF:type Fresnel:Lens ;
    Fresnel:instanceLensDomain "SELECT ?x WHERE ?x ontExample:knows ?y " 
    "Fresnel:sparqlSelector.
```

In case of a Lens has as domain value the `Fresnel:classLensDomain` value, this Lens needs to define what are the properties that should be displayed by using the `Fresnel:hideProperties` property and what is the order in which these properties should be showed by using the `Fresnel:showProperties` property.
2.3. Fresnel

The value of `Fresnel:showProperties` and `Fresnel:hideProperties` can be either a single property selector or an ordered list of property selectors. The Lens should be define also the `Fresnel:purpose` property to help browsers to decide which Lens to use in case of conflict between two or more Lenses that can be applied on the same resources. The Lens purpose depends on the application domain and the browser adopted; e.g. each browser can define its own customized "Fresnel:purpose" value, and can give maximum priority to the Lenses that have this value on their "Fresnel:purpose" property or also for a particular domain it is possible determine a customized "Fresnel:purpose" value that identify the Lenses with maximum priority. Moreover, Fresnel provides an open interface to create domains or browser-specific purposes and defines only two general purposes that might be useful across all browsers:

- The `Fresnel:defaultLens` purpose is intended to indicate a default Lens for a specific class.
- The `Fresnel:labelLens` purpose indicates a Lens that provides a short, human-readable label for an instance of a specific class.

2.3.2 Format

The Format determines how to render the resources, their properties and values. The format vocabulary also provides a link between RDF and styling languages such as CSS [css(1994)], which are used to specify styling attributes like: colors, fonts and borders through external stylesheet. Formats describe the representation formalism of RDF resources properties and instances by defining:

- How the resources are classified.
Chapter 2. Methods for Information Visualization and Knowledge Visualization

- How the resources values are displayed.
- Which additional contents, like commas and periods, will be displayed between and after resources values.
- What is the references to CSS classes which define fonts, colors, borders and other styling attributes.

In Formats it is possible to define the Fresnel:propertyFormatDomain property. This property specifies the set of properties (i.e. RDF property) on which the format is applicable. A format can have more than one format domain. Some example of formatting of properties are:

- Displaying the values of the properties as an image without a label.
- Displaying the labels of the properties using the Arial font in 12pt size.
- Displaying the values of the property with the custom label as example: “This person ‘s friends: ”.

In the algorithm it is showed an example of format:

**Algorithm 5 Fresnel Format**

```
:knowsFormat RDF:type Fresnel:Format ;
   Fresnel:propertyFormatDomain ontExample:knows ;
   Fresnel:label "This person knows the following people: "xsd:string .
```

To formatting RDF instances, in Formats it is possible to define, in a similar way to Lens domains, the Fresnel:classFormatDomain or Fresnel:instanceFormatDomain. FSL or SPARQL expressions can be used as values of Fresnel:instanceFormatDomain.

Some use cases for formatting instances are:
2.3. Fresnel

- Drawing a border around the area used to display an instance.

- Specifying Formats that only apply to a custom set of instances. Example: display all persons that are younger than 21 using a special color.

- Handling mixed content, where a specific format should be applied depending on the RDF:type of a property value. Example: in a calendar, private activities should be displayed with a green background, work activities with a red background.

Furthermore it is possible to explicitly link a Lens and a format with Fresnel:use as shown in the example below:

**Algorithm 6 Explicit Link**

```
:ontExamplePersonLens RDF:type Fresnel:Lens ;
    Fresnel:instanceLensDomain "SELECT ?x WHERE ?x ontExample:knows ?y " ^ ^ Fresnel:sparqlSelector ;
    Fresnel:use :knowsFormat.
```

2.3.3 Fresnel: purposes and uses

As mentioned above main purpose of the definition of Fresnel language is to give the possibilities to define through a standard language all the information about the RDF visualization. Since its definition, several tools adopt Fresnel as standard language to define visualization information as example: Semantic Web browser (further details in 2.4.1.1), RDF editors etc..

The following section analyzes the Knowledge-Based Visualization Systems and examines the role of the Fresnel vocabulary in this kind of systems.
2.4 Knowledge-Based Visualization Systems

Recently many approaches with the aims to visualize RDF data have been developed, in [Aba-Sah Dadzie and Matthew Rowe(2011)], the authors describe many different approaches to visualize them, they show how the research field of data visualization is a continuously growing interest.

As mentioned above the visualization process can be divided in three steps: selection, structuring and formatting. While the selection deals with the identification of relevant sub-graph in a RDF graph (see chapter 3), the structuring and formatting steps deal with the organization and styling of data.

Following examples of existing visualization approaches show some different implementations of the structuring and formatting steps in the Knowledge-Based Visualization systems:

- Graph-based: provides a graph view representation.
- Faceted browsing: provides forms view filled with formatted RDF representation of resources.
- Domain-specific: provides a graphical representation tailored on a specific domain.
- Widgets-based: provides a rich graphical representation of resources using graphical metaphor (e.g. table, list etc.. in XHTML, XUL, SVG etc) and stylesheet.

In the figure 2.5 it is shown an example of graph view. As shown in figure 2.6 this kind of visualization lacks on scalability, indeed for very large graphs, the graph-based
visualization approach is not very understandable.

![Figure 2.5: OWLViz graph view of a small Ontology.](image)

![Figure 2.6: OWLViz graph view of a big Ontology.](image)

In the figure 2.7 it is shown an example of Formatted RDF forms provided by marbles [Becker and Bizer(2008)]. In the figure 2.8 it is shown an example of Domain-specific visualization provided by Talis Research Funding Explorer.

Finally in figure 2.9 is shown, as example of Widgets-based approach, a table view.

---

2http://data.gov.uk/apps/research-funding-explorer
Chapter 2. Methods for Information Visualization and Knowledge Visualization

Figure 2.7: Marbles Formatted RDF display view.

provided by the Tabulator [Tim Berners Lee et al. (2006)] tool.

2.4.1 State of the art of RDF browsers

In the Semantic Web the metadata are the components that are able to implement the assumption of "data about data", to implement this vision RDF has been defined, RDF (Resource Description Framework) is a language for describing metadata in the Semantic Web.

One of the main peculiarities of traditional Web browsers is that they can work with any content, providing that it is specified according to some given standard they accept: RDF Browser do the same, by complying with Semantic Web paradigm of
2.4. Knowledge-Based Visualization Systems

Figure 2.8: Domain-specific visualization.

<table>
<thead>
<tr>
<th>Candidate Recommendation</th>
<th>Candidate Recommendation deliverer</th>
<th>Candidate Recommendation deliverer chair</th>
<th>Candidate Recommendation implementation feedback due</th>
<th>Candidate Recommendation deliverer endingDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSS Mobile Profile 1.0</td>
<td>Cascading Style Sheets</td>
<td>Bert Biss</td>
<td>2003-01-25</td>
<td>2005-09-30</td>
</tr>
<tr>
<td>CSS Print Profile</td>
<td>Cascading Style Sheets</td>
<td>Bert Biss</td>
<td>2004-08-25</td>
<td>2006-09-30</td>
</tr>
<tr>
<td>CSS3 Basic User Interface Module</td>
<td>Cascading Style Sheets</td>
<td>Bert Biss</td>
<td>2004-11-11</td>
<td>2006-09-30</td>
</tr>
<tr>
<td>CSS3 Paged Media Module</td>
<td>Cascading Style Sheets</td>
<td>Bert Biss</td>
<td>2004-09-25</td>
<td>2006-09-30</td>
</tr>
<tr>
<td>Media Queries</td>
<td>Cascading Style Sheets</td>
<td>Bert Biss</td>
<td>2003-01-31</td>
<td>2005-09-30</td>
</tr>
<tr>
<td>SPARQL Query Language for RDF</td>
<td>RDF Data Access Working Group</td>
<td>Kendall Clark</td>
<td>2006-06-06</td>
<td>2006-01-31</td>
</tr>
<tr>
<td>SPARQL Query Results XML Format</td>
<td>RDF Data Access Working Group</td>
<td>Kendall Clark</td>
<td>2006-06-06</td>
<td>2006-01-31</td>
</tr>
<tr>
<td>Web Services Choreography Description Language Version 1.0</td>
<td>Web Services Choreography Working Group</td>
<td>Steve Rose-Tallbot</td>
<td>2006-03-31</td>
<td>2006-12-31</td>
</tr>
<tr>
<td>Web Services Choreography Description Language Version 1.0</td>
<td>Web Services Choreography Working Group</td>
<td>Martin Chapman</td>
<td>2006-03-31</td>
<td>2006-12-31</td>
</tr>
<tr>
<td>Web Services Description Language (WSDL) Version 2.0 Part 0: Primer</td>
<td>Web Services Description Working Group</td>
<td>Tony Rogers</td>
<td>2006-07-01</td>
<td>2007-01-31</td>
</tr>
</tbody>
</table>

Figure 2.9: Widgets-based.
reusability and sharing of information. There are many works on RDF Browsing in literature (many of which have lead to the realization of prototype tools). All of them, propose even really different strategies for browsing RDF data.

Some tools provide nested boxes layouts, as Haystack and Tabulator, that is, recursively contained boxes of property value pairs.

Other kind of tools provide link navigation with facets. Facets are different dimensions, perspectives, of the underlying data. Often, the values of this dimension are hierarchically structured to represent relevant categorization of data driven by each perspective. Facets are used in many different RDF Browsers like Longwell\textsuperscript{3} and /facets\textsuperscript{4}.

Finally another class of tools provide a graph representation such as:

- RDFSVis, is a visualization service for ontologies represented in RDF schema;
- OwlViz, is an highly configurable ontology visualization tool integrated in Protégé;
- IsaViz, is a flexible tool for RDF graph visualization, with a number of functions for zooming, editing, searching and browsing the graph structure;
- RDF Gravity, is a tool to visualize and navigate directed graphs built in RDF and OWL, with the possibility to zoom, search, filter out and visualize specific parts of RDF graphs;
- Cluster Map, is a key component of the Spectacle system\textsuperscript{5} used for the visualization of ontological data, with a very expressive and configurable interface;
- GVis is a general purpose, flexible and highly customizable graph visualization

\textsuperscript{3}http://simile.mit.edu/wiki/Longwell
\textsuperscript{4}http://db.cwi.nl/rapporten/abstract.php?abstractnr=2037
\textsuperscript{5}http://spectacle.aidministrator.nl/
2.4. Knowledge-Based Visualization Systems

tool is, used in the context of the Hera project [Richard Vdovjak et al. (2003)] for visualizing large RDF graphs;

• Semantic Turkey [Maria Teresa Pazienza et al. (2011)], is a Knowledge Management and Acquisition tool, providing graph exploration of edited ontology.

RDF graphs may not be intuitive to understand, in particular when they are very large and the relationships between its concepts are numerous. Moreover, though quite self-explicative, also RDF needs some non intuitive constructs to represent its data, such an in the representation of n-ary relationships, which needs to reify relationships and constructs chains of blank nodes which need to be properly interpreted (and thus shown accordingly).

2.4.1.1 Fresnel Based RDF Browsers

Last generation RDF Browsers exploit the Fresnel vocabulary to define patterns of representation, some examples of Fresnel Based RDF Browser are:

• LENA (Universität Koblenz-Landau http://code.google.com/p/lena/).

• Longwell (http://simile.mit.edu/wiki/Longwell).

• IsaViz (http://www.w3.org/wiki/IsaViz).

• Horus (Freie Universität Berlin)

• OAT: OpenLink AJAX Toolkit (OpenLink Software)

• Haystack (haystack.lcs.mit.edu).

• Tabulator (http://semanticweb.org/wiki/Tabulator).
Chapter 2. Methods for Information Visualization and Knowledge Visualization

- LESS (http://less.aksw.org/help).

In general RDF browsers compliant with Fresnel require to be setup with one or more Configuration Files realized by experts of the considered domain. This files contains specification for all the Lenses that will be applied on data and their related Formats.

The rationale behind Fresnel relies in applying to Lenses and Formats the same Semantic Web principles of openness, shareability and reuse that are applied to knowledge resources, by creating reusable pairs of ontologies/Configuration Files, which can be searched, browsed, filtered according to user specific needs, downloaded and finally applied to local browser.

### 2.4.2 Problems and open challenges in Knowledge-Based Visualization Systems

Each of the categories of Knowledge-based visualization systems mentioned above presents some problems that should be solved.

- Graph representation has some problems in displaying a very large graphs.
- Faceted browsing lacks the possibility to show clearly the relationship.
- Domain Specific has obvious limits concerning the possibility to display data coming only from a specific domain.
- Widgets-based is a good solution while the existing implementation requires definition of Configuration Files. Most of the attempts to implement this kind of tools (see 2.4.1.1) foresee the manual configuration from users.
A worthwhile solution to this problems is the realization of a Knowledge-based visualization systems able to generate a Widgets-based GUI through the automatic creation of the Configuration Files.

2.5 Conclusions

As shown in the previous sections existing knowledge-based visualization approaches have several limits and problems.

 Among the reviewed approaches, the widget-based approach seems to be the closest to achieve the creation of knowledge-based visualization. To accomplish this challenge this kind of approach should be able to work with any content and should allowing to the users to interact with RDF data without settings. A worthwhile solution to the settings problem, seems to be the definition of an approach that realizes the automatic generation of the Configuration Files. In this way common users will be able to browsing different kind of RDF data without settings. We think that the Fresnel vocabulary is the best candidate to become the standard language in the creation of the the Configuration Files. There are many reasons that support this assertion:

- The Fresnel vocabulary is an extension of RDF, this means that it is based on the same standard that is used to represent the metadata. So the Fresnel data, may be stored in the same knowledge-base in which metadata are stored.

- If the Configuration Files are produced through Fresnel vocabulary, they can be modified with the same API that are used to interact with metadata.

- The Fresnel vocabulary is a W3C standard, this peculiarity allows the reusability of the Configuration Files, among RDF Browser that adopt the same standard.
Chapter 2. Methods for Information Visualization and Knowledge Visualization

As described in section 2.3, the Fresnel vocabulary is a standard that defines three kinds of selectors. Between them the SPARQL Selector is the selector that introduces the possibility to integrate SPARQL in the query contained in the Lenses. As previously mentioned, retrieving of metadata, is a challenging task in the Knowledge-Based visualization research field, we think that the use of SPARQL in Lenses can give an improvement to this systems especially when queries are automatically generated. The use of the SPARQL selectors in Lenses allows to adopt the standard query language to RDF in the Configuration Files. For this reason, we affirm that to adopt the SPARQL selectors into the Configuration Files is a worthwhile solution.

In the next chapters we describe in details (chapter 3 and chapter 4) the approaches to querying metadata and the methods to identify relevant subgraphs. In particular in chapter 3 we analyze the Semantic Search research field and we describe the different methods to create the SPARQL queries and in chapter 4 we review the Web page visual analysis techniques.
As mentioned in the previous chapters the first step of visualization process is the selection of the data. This chapter is about the methods to search data on the Semantic Web. Section 3.1 describes the peculiarities of the Semantic Search and in particular the features of the Semantic Web Data Search. Section 3.2 discusses existing approaches to create SPARQL queries. In particular we analyze the approaches that exploit knowledge to expand or to guide the formulation of queries that are able to interrogate Knowledge repository. Section 3.3 analyzes the Semantic Search approaches that are adopted in the RDF Browser.

### 3.1 Semantic Search Approaches

The term Semantic Search has been traditionally adopted with two meanings, according to the interpretation of these words in different research communities: on 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. As in [Christoph Mangold(2007)](https://doi.org/10.1007/11697662_2), the 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
Chapter 3. Semantic Search

with the retrieval of Semantic Web data.

In the traditional search engines, the search process is independent from the real users needs. Indeed, it is realized by the implementation of an algorithm that identifies mere syntactically association from keywords introduced by the users and keywords present in Web data, this approach affects precision and recall of search algorithm.

The main peculiarity of Semantic Web is the introduction of the machine-readable data, this makes possible the implementation of search algorithms able to understand the meaning of submitted queries.

Many approaches following this second interpretation were published in these last years, embracing several application areas and exposing different realizations. In this thesis we pay more attention to the second interpretation, we will provide more details regarding the approaches and tools that implement the second interpretation in the following sections.

3.2 Query Formulation

One of the main challenges of the Semantic Search research field is the creation and solving of the semantic queries.

These queries are able to select relevant data from the Semantic Web data, but their composition is not a simple task. While in the traditional search engine the queries are a simple list of keywords linked by simple logical relations (as example “and”, “or ”, “not ” etc.), the Semantic Search, instead, needs the submission of semantic queries that are structured as a graph of concepts and relations. This implies a problem in the semantic search, indeed, while a list of keywords can be provided from all kind of users, formulation of semantic queries is not an easy task and not all users are able
3.2. Query Formulation

to carry it out. To formulate a semantic query, users need to know a Semantic Query Language (i.e. SPARQL) and also they need to have some knowledge of the domain.

One of the challenges of the implementation of the RDF Browsers is to implement visual interfaces that allow common users to submit the semantic queries hiding them the complexity. Many different approaches are provided to achieve this challenge. In particular we can distinguish three kinds of approaches:

- Manual approach: asks users to submit manually the semantic query and provides only some utilities to make easy the editing of the queries.

- Simple Keyword query approach: asks users to submit a simple keywords query and than tries to understand the meaning of these keywords using some semantic based algorithms. First advantage of this approach is the possibility to hide to the users the complexity of the semantic queries, but differently from previous one it is prone to some error in interpretation of the queries.

- Graphical interface approach: provides a graphical interface to allow semantic query formulation. This approach leverages the power of the graphic representation, to allow users to compose the queries by the visual composition of queries.

As mentioned above, to formulate the semantic queries it is necessary to known a semantic query language. SPARQL [Eric Prud’hommeaux and Andy Seaborne(2008)] is the W3C standard language to formulate the semantic queries. An overview of the SPARQL query language will be provided in the following section 3.2.1.
Chapter 3. Semantic Search

3.2.1 SPARQL Query Language

SPARQL [Eric Prud’hommeaux and Andy Seaborne(2008)] is a Query Language for RDF [Ben Adida and Mark Birbeck(2008)]. RDF is a direct, labeled graph data format for representing linked data.

As SQL is used to retrieve view of data in a database, SPARQL is used to identify relevant subgraph in a Knowledge base.

SPARQL has several query forms:

- SELECT returns resources selected using a clause (often the clause is WHERE).
- CONSTRUCT returns an RDF graph created by substituting variables in templates.
- ASK returns a boolean indicating if a query pattern matches or not.
- DESCRIBE returns an RDF graph that describes the resources found.

In SPARQL queries it is also possible to specify the RDF dataset to be used for the matching, by using the FROM clause and the FROM NAMED clause, to describe this. The FROM and FROM NAMED clauses allow query to specifies an RDF dataset by reference; this means that the dataset should include graphs that are obtained only from RDF datasets identified by the given IRIs \(^{1}\) (i.e. the absolute form of the given IRI references). The dataset resulting from a number of FROM and FROM NAMED clauses is:

- a graph consisting of the RDF merge of the graphs referred to in the FROM clauses,

\(^{1}\)http://www.w3.org/TR/2004/REC-RDF-concepts-20040210/#ref-iri
• a set of pairs, one from each FROM NAMED clause.

To filter results of a SPARQL query in the SPARQL language the FILTER constraint has been defined. The FILTER constraint is used in a query to filtering results, filter can be expressed by using a regular expression, or by introducing a simple logical condition. Furthermore results can be managed using one of these query modifiers:

• Order: orders results ;

• Projection: selects only certain variables ;

• Distinct: removes duplicate results from the solution ;

• Reduced: removes some non-unique results;

• Offset: removes first offset-1 results from the solution;

• Limit: restricts the number of results at the first limit results.

### 3.3 Semantic Search in RDF Browser

This section describes the approaches of semantic query formulation, in particular section discusses different implementation of these approaches in the RDF Browsers. In general a RDF Browser should provides some features for interrogate its own RDF repository, this because users that navigating in large graphs of resources, may have different information needs and queries are the best way to enable him to express them.
3.3.1 Manual Query Composition

As mentioned above, manual query composition is implemented by many RDF Browsers as Protege’, TopBraid Composer\(^2\) and Semantic Turkey (see figure 3.1).

![Semantic Turkey SPARQL Panel](image)

Figure 3.1: Semantic Turkey SPARQL Panel

These kind of RDF browsers are often used by experts in knowledge management because they are able to compose semantic query, however it is worthwhile to implement some visualization features which can help users to instantly and intuitively understand why and how the results were retrieved.

For instance: TopBraid Composer\(^3\) provides a SPARQL Web Pages\(^4\) support\(^5\) to construct Web pages filled with the SPARQL queries result’s.

As shown in the figure 3.1 the Semantic Turkey system provides a SPARQL panel to submit SPARQL queries.

\(^2\)http://topbraidcomposer.info/
\(^3\)http://topbraidcomposer.info/
\(^4\)http://uispin.org/
\(^5\)http://topquadrant.com/topbraid/uispin/
Finally Protege has a built-in SPARQL query panel as described in the site\footnote{http://protege.stanford.edu/doc/SPARQL/}.

This kind of tools are used by expert users able to create manually semantic queries and to edit ontologies, indeed other approaches to create semantic queries have been implemented to allow common users to submit the RDF queries, further details about these approaches will be provided in the following paragraphs.

### 3.3.2 Semantic Query Expansion

In \cite{Tran et al.(2007)} the authors present an approach which translates a keyword query into a DL conjunctive query which can be evaluated with respect to the underlying knowledge base (KB). To map keywords into concepts of an ontology, first of all they define a generic approach for Ontology-Based query interpretation specifying the two following assumptions:

- **Ontology-Mental Correspondence**: this assumption defines an entity-wise and a structural correspondence between the mental model $O_u$ (i.e. user needs) and the system resource model $O_s$ (i.e. ontology). This means, that the elements and the associative structure in $O_u$ correspond to the ontology entities and to the structure in $O_s$, respectively.

- **Locality of Information Need**: In this assumption authors affirm that the $O_s \subseteq O_u$ and the entities in the $O_u$ are connected over a maximum distance $d$.

  This means that for any pairs of ontology entities $a, b \in O_s$ there is or a direct connection $\langle a, b \rangle$ or a sequence of $x_i$, such that $a = x_0$ and $\langle x_0, x_1 \rangle$, $\langle x_1, x_2 \rangle$, ..., $\langle x_{n-1}, x_n \rangle$, and $\langle x_n, x_b \rangle$ and $n < d$.

  If are presents different sequences that connect the two entities $(a, b)$, authors
assume not only that there is a maximum distance between entities (i.e. d), but also that the connections over smaller distances are more likely to contain the information that the user looks for.

Based on these assumptions authors present a generic approach to translate an user query in a semantic query which consists of three steps.

First, the keywords in the user question, are mapped into ontology elements by using the system resource model $O_r$. Second, further ontology elements are explored, to better cover, the initial information needs in the mental model $O_u$. Finally, from this more refined ontological representation of the needs, the semantic query will be derived.

Many other works that describe different approaches to map keywords to ontological concepts has been published, for example in [Yuangu Yi et al. (2006)], authors present SemSearch. While in the previous approach keywords are mapped only into concepts of an ontology in this work the authors try to improve the search performance by expanding the query also with instances and relations. In SemSearch, the keywords are interpreted as either instances, concepts or properties, respectively. This means that for a query composed by two keywords there are nine possible templates to be instantiated. Different from previous approach in SemSearch authors assume that entities denoted by keywords can be connected through a direct relation in the ontology. This work give a very important contribute to understand users information needs and refine queries to get more precise results.

However, determine meaning of keywords it is not enough to completely understand the users needs. Furthermore, the mapping process of keywords fails, if the keywords introduced by the users do not correspond to the labels of the concepts to be
recognize. Identify the semantic relations of metadata seems to be a good solution to support the retrieving of information which are closely related to the keywords. For these reasons recently, answering keyword-based queries on graph-structured data has emerged as important research topic, to solve the problems of identify not only the meaning of the keywords but also their relations.

An algorithm for the exploration of top-k matching subgraphs has been presented in [Thanh Tran et al. (2009)]. In this works the authors improve the keywords mapping, through the implementation of an algorithm that supports imprecise matching. This algorithm incorporates syntactic and semantic similarities, then the users does not need to know the labels of the data elements, when they make a keywords search. The authors make a ranking of the possible matching sub-graphs and provide a list of top-k matching sub-graphs. The ranking task has been carried out through the application of an algorithm that calculate the matching probability between a sub-graph and the user query.

As mentioned above this algorithm provides a probabilistic approach, then it is prone to mistakes in interpretation of the users queries.

3.3.3 Visual Query Composition

As shown in previous paragraphs, both manual query composition and semantic query expansion, have some criticality. While manual query composition requires that the user is experienced both in the domain and also in the creation of semantic queries. The query expansion approach instead, is prone to a certain approximation and may also lead to a wrong interpretation of users needs.

Recently a new approach to compose queries has been proposed by different RDF
Chapter 3. Semantic Search

Figure 3.2: Tabulator Visual Query Interface.

Browsers, the visual query composition. This approach allow users, that are not experienced in composition of the semantic queries, to create theirs queries, by using visual interface.

vSPARQL [Shaw et al. (2010)] is a set of extensions to the SPARQL query language, it allows the creation of view definitions over RDF data. This language makes possible the extraction, modification, and augmentation of RDF data.

Another attempt to encourage visual query composition is the definition of IML [7]. It is a data flow graph transformation language for manipulating RDF data.

In the figure [3.2] it is shown a screenshot of tabulator visual query composition panel, in this panel, users can select a node and expand it, to view all fields of this node. Then users can highlighting one of these fields, and tabulator create a query that is able to select all resources that have the same field value and display them.

Furthermore Tabulator, allow users to refine the created queries. This is performed or trough the visualization of SPARQL code and its modification or trough the iterative selection of fields, in the tabulator visual query interface.

Finally users can ask tabulator to show results in a tabular forms as shown in the figure.

3.4 Conclusions

Further notable examples of visual query composition are:
NITELIGHT [Alistair Russell et al. (2008)] system, OntoVQL [Fadhil and Haarslev (2007)],
SEWASIE [Catarci et al. (2004)], SPARQLViz that is a plugin for IsaViz provided by
Emmanuel Pietriga\(^8\) and finally iSPARQL\(^9\).

3.4 Conclusions

Previous approaches present many open challenges: manual query composition lack
on common users usability, Semantic Query Expansion can become liable for errors in
interpretations of users needs. In conclusion Visual Query Composition seems to be
the best solutions to this problem because asks users to interact with a visual interface
that hides the complexity of the queries and overcame the approximation of Semantic
Query Expansion. This approach however may be not satisfactory when users need to
submit several and/or complex queries. Furthermore 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 understand intuitively why and how the
results were retrieved.

\(^8\)http://sparqlviz.sourceforge.net/index.php
\(^9\)OpenLink iSPARQL, http://demo.openlinksw.com/isparql/
Chapter 3. Semantic Search

In [Myungjin Lee et al. (2010)] Myungjin Lee presents a semantic association-based algorithm and shows how to provide proper visualization and navigation methods for the results. The author affirms that to accomplish the challenges of select data and to show them in a charming manner, you should implement an approach able to implement both a semantic search algorithm and a graphical representation. Also he affirms that this approach should not requires that users are experienced in knowledge domain or in semantic technologies.

In this thesis we propose an approach (see chapter 5) to implement a system that is able to interact with common users, in this approach we provide both a semantic search algorithm (see 5.6) and an automatic GUIs generation system. Our approach is able to generate a complete set of semantic queries and related pattern of visualization.
Classification of Web content is an essential task in many information retrieval tasks. The retrieval of Semantic Web data presented in chapter 3 can be improved using the Web page segmentation algorithms. This chapter is about the methods to improve semantic search using the Web page visual segmentation algorithms. Section 4.1 describes the segmentation approaches. Section 4.1.3 describes in details the Web page visual analysis techniques. Section 4.2 describes the improvements that the Web Page visual analysis leads to the Semantic Search approaches. Finally section 4.3 draws the conclusions about Web page visual analysis.

4.1 Web page segmentation approaches

There are three main approaches to perform Web page segmentation:

- **DOM based segmentation approach**: analyzes DOM tree nodes.
- **Template Detection approach**: looks for standard templates.
- **Visual analysis approach**: analyzes the visual forms of a page.

Peculiarities and differences between these approaches will be described in the following paragraphs.
4.1.1 DOM based segmentation Algorithms

The DOM based segmentation class of algorithms analyzes the structure of the DOM tree, and its nodes, in term of content and tree related features. Also some basic visual features are analyzed with the intention to identify nature of DOM nodes. Examples of DOM-based algorithms can be found in [Vineel(2009)]. In this works the authors characterize the nodes of DOM tree structure, based on their content size and entropy. Based on this characterization of DOM nodes, authors develop an unsupervised algorithm to automatically identify segments of a given Web page.

4.1.1.1 The Document Object Model

The Document Object Model (DOM) is a standard W3C since 1998. It is a platform and language-independent model that allows programs and scripts to dynamically access and update the content, structure and style of documents. By using DOM standard it is also possible after processing of documents to incorporate the results of this processing back into the presented page.

It is especially used in area of World Wide Web as a means to work with XML, XHTML and HTML documents. It is the most common model used for this purpose. Many API have been developed to interact with DOM documents for many different program languages (e.g. java, javascript ...)

4.1.2 Template Detection algorithms

The Template Detection algorithms are based on the presence of templates in a determined kind of pages, templates are pre-determined visual structures used in modern

\[\text{http://www.w3.org/DOM/}\]
4.1. Web page segmentation approaches

websites to create Web pages. This concept brings easier Web page creation and organization, therefore it is mainly used in large CMSs, e-shops and blog applications.

In general these methods foresee a clustering step to divide Web pages in clusters. Each cluster represents a Group of pages that adopt the same template. Thanks to the clustering step, the template detection methods usually scale better than the algorithms described above. This because instead of segmenting one page at a time, more pages are analyzed in parallel and the result is then applied to all of them. A key aspect of the template detection algorithms is that they recognize in pages only in two parts: template and content. The template contains all information besides the content and is not segmented further, usually templates are discarded and only the contents are analyzed.

In [Vieira et al. (2006)] authors describe a method for Web page template detection, that identifies templates and removes them from Web page. The algorithm is composed of two phases first templates are recognized in a small Group of pages and then the templates detected are removed from the remaining pages in the collection. Templates are removed because authors affirm that they adversely affect on retrieval process. In the following section a discussion, about the possible improvement of semantic search process exploiting the Web page visual analysis results, will be presented.

4.1.3 Web page visual analysis Approaches

Each Web page has two representations. One is the text representation written in HTML. The other one is the visual representation rendered by a Web browser. They provide different views of a page.

Many approaches of Web page analysis are focused on the text representation while
The Web page visual analysis is about the realization of algorithms able to divide a Web page in a series of meaningful blocks, exploiting the visual representation, for identify limits of each of this blocks.

Web page visual analysis algorithms can be viewed as a complement of Semantic Search research field, because they are able to support and improve performances of Semantic Search algorithms. A particular field of research called SEO (i.e. search engine optimization) has been introduced to indicating all Web page segmentation approaches that aim to support search algorithms, in this field of research, Web page visual analysis, is one of the most interesting topics.

In [Shipeng Yu(2003)] the authors introduce a Web page segmentation algorithm (VIPS) that attempts to divide the page into meaningful blocks based on visual representation (see later section for details).

### 4.1.3.1 VIPS algorithm

In this section the VIPS (i.e. Vision Based Page Segmentation) algorithm is described in details. Many algorithms of visual page segmentation are based on VIPS approach, so a description of this approach it is useful to understand most of the recent works and to underline the improvement that a segmentation algorithm can give to the Semantic Search. The VIPS goal’s is to divide the input Web page in: a set of blocks, a set of separators and a set of relations between these blocks.

To define the blocks, VIPS ensures that blocks are not overlapping. This means that their union creates their parent block.

A very important feature of VIPS is that each block, is recursively segmented, and is
4.1. Web page segmentation approaches

represented by a further set of: blocks, separators and relations. After the application of VIPS algorithm, the page can be represented as a tree-like structure that contains all blocks recognized. The root of this tree structure is the Web page itself. The leaf nodes of the tree structure, are called basic objects. Each basic object corresponds to one node in the DOM tree. Hence each block can contain one or more nodes of the DOM tree. It is important to underline that the tree structure of blocks and the DOM tree don't have to correspond, i.e. the visual blocks do not need to correspond to a node in the DOM tree.

For this reason, recently works (as [Liu et al. (2010)]) provide additional heuristics and classification algorithms. They may be carried out on segmentation results with the aim to generate a mapping between DOM and visual tree. This additional information can be stored together with the output of VIPS algorithms to provide a more exact information about blocks. For instance using these methods it is possible to identify for each block some information about its position and size. Furthermore these properties can be expressed as absolute numbers or relative to the parent block, then it is possible to use the alignment between block and its parent to get more useful information regarding the meaning of block itself.

To measure the alignment from blocks and DOM node, VIPS algorithm defines a degree of coherence, that indicates the degree of coherency of the block, this is calculated by using a strategies based on the blocks visual property. The Degree of Coherence has the following properties:

- More the value of the Degree of Coherence is high, more the content within the blocks is consistent;

- In the hierarchy tree, the Degree of Coherence of the child is not smaller than
that of its parent.

In VIPS, the Degree of Coherence is expressed through an integer that is contained in a range (from 1 to 10), although alternatively different kinds of ranges (e.g., real numbers, etc.) could be used.

In addition to blocks very significant is the identification of the separators and their peculiarities as an example: width or visibility defined by borders or background color of adjacent blocks. Separators are horizontal or vertical lines in a Web page, that visually do not cross to the blocks in the page. From a visual perspective, the separators are good indicators for discriminating the different topics within the page. The weight of a separator is determined using an approach based on properties of its neighboring blocks.

Determine relations between blocks is the last operation performed by the VIPS algorithm. The relations between blocks can be defined as the degree of visual similarity of blocks in relation. A particular attention must be given to the identification of relations between blocks of the same level of the tree. This kind of relation is established between two blocks that are adjacent. These relations are very significant to identify the visual blocks (i.e., two blocks that are similar could be grouped into the same block).

### 4.1.3.2 Other Web page visual analysis approaches

In literature many others algorithms have been presented, many of these algorithms are based on visual Web pages analysis (VIPS). As example:

Burget in his work [Burget(2007)] focuses on similar problems as VIPS, but structures described in this work are slightly different. The tree produced by this algorithm
4.2 Using Web Page Analysis to improve Search Engine Performance

contains two node types: visual areas and content nodes.

The authors of [Li(2010)] propose an improved algorithm of vision-based page segmentation. In addition to the traditional approach, the authors cluster the blocks with similar content and spatial structures. The objective of this work is to identify informative clusters and to discard noisy blocks (i.e. non-informative blocks).

4.2 Using Web Page Analysis to improve Search Engine Performance

Retrieval of information in a Web page is a more difficult task respect to traditional document search, because a Web page is not a good information unit in which to search information.

One of the main problems of traditional information retrieval in Web page, 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 problem [Wu and Zeng(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.

Microsoft, for example, has patented VIPS in relation to this algorithm its chief researcher affirms: "VIPS presents an automatic top-down, tag-free independent approach to detect Web structure. It simulates how a user understands Web layout structure based on his visual perception."
Google received a patent for its method of identifying of the visual gaps in a page, with the original filing dating to 2006.

Yahoo filed a patent application in 2008 for a search algorithm that looks to visual cues instead of the HTML structure of a page to discern the semantic content of a page.

The goal of these approaches is that the relevance of terms within a page becomes a function of its perceived significance given its position within the visual presentation, irrespective of the HTML structure of the document. Exploiting these results in search engines it is possible to weigh the links through an approach based on their prominence and location within the page's semantic presentation.

Into the Semantic Web, understanding the meaning of the information contained in a Web page is one of the main issues. The Web Page Segmentation approaches can be used to facilitate the comprehension of data in a page. In particular the possibilities to recognize the visual blocks can be used to discover further information respect content of these blocks and to improve retrieval of relevant information. In general the visual organization of a page can be interpreted as the logical schema of the data. Usually the person who creates the schema of a Web page, has a good knowledge of domain of data, hence he creates the structure of the Web page based on content of data. The analysis of the visual organization of data can be considered as a worthwhile preprocessing task before Semantic Search task.

In the past, many works have suggested to use the segmentation as preprocessing task, able to remove the templates and to extract only the content of a page, but recently and in particular in the Semantic Web vision, this approach has been replaced by the integration of information provided both by data and schema of a Web page.

Furthermore in the Semantic Web, the Web page segmentation preprocessing led
improvement in many other fields of research, as example in [Petasis et al. (2008)] authors propose an approach to Segmenting HTML pages using visual and semantic information, to improve the automatic semantic annotation of the Web pages.

4.3 Conclusions

As mentioned above all previously approaches present some problems:

The main problem related to DOM based segmentation approach is that it do not makes any assumption regard the meaning of visual organization of a Web page.

The main problem of Template Detection approaches, instead is that they recognize only two kinds of information template and content, template contains all information besides the content and no assumption regard its composition is provided. That is why these algorithms cannot be used as a complete replacement of vision-based segmentation algorithms.

While vision-based segmentation algorithms seem to be the worthwhile solution to carry out the visual segmentation of the Web pages, they present some problems. One of these is related to the performance in terms of speed of execution of the algorithm. Another one is the lack of precision of these kind of algorithms. The precision problem might be solved through the introduction of more complex algorithms but this solution could affect the performance of the algorithm.

Finally one of the main limits related to the integration between VIPS algorithms and semantic-search algorithms is that the recognized basic objects (i.e. leaf nodes of the visual tree) are be considered as a black boxes. This means that the comprehension of the meaning of basic objects is completely delegated to the semantic search algorithms. To solve this problems it is possible to investigate the visual composition of each of
Chapter 4. Web page visual analysis

these objects to identify relevant visual elements that compose them. Another key aspect is to identify the relation between these visual elements, to make a sort of “visual tree of visual tree basic objects”

In order to achieve the challenge of determine the meaning of the blocks, we propose a new VIPS approach (see section 5.5). Our approach aims to implement a Web page visual analysis algorithm that introduces a further level of segmentation of the basic visual objects, and a more refined detection process of the relations between the visual elements detected by our algorithm.
Part III

Thesis Contributions
This chapter main focus is on Semi-Automatic GUI Generator system. To the best of our knowledge, the Semi-Automatic GUI Generator (SAGG from now on) approach is new in knowledge-based visualization research field. In section 5.1 we describe the overall idea behind SAGG. In section 5.2 we discuss the scenario in which SAGG has been developed. In section 5.3 we analyze the model on which SAGG is based. In section 5.5 and 5.6 we describe the algorithms developed to implement the SAGG’s approach and their interaction. We describe the architecture of SAGG in section 5.7, the characteristics of each modules have been described in the following subsections. Section 5.8 shows an use case of SAGG system. In conclusion in section 5.9 we provide a detailed evaluations of SAGG’s approach.

5.1 Overall Concept

The layer-cake of technologies and languages for information representation in traditional Web content identifies well-separated levels of competence where artistic work, content development and technological aspects may be assigned to the most appropriate figures. Styles (e.g. CSS), content structure (HTML tags), embedded data (e.g. and RDFa [Ben Adida and Mark Birbeck(2008)], server-side and client-side technologies for dynamic content publishing provide different levels of abstraction where all of the
above figures has a specific role. This clear separation has led to highly specialized
development tools allowing management of the aspects of interest for each figure, but
also the proper abstraction from the other layers, and the simplification required for
their competencies. In the same way, the RDF Browsers should provide a user inter-
face that binds the graphical structure to metadata, and allow ontology experts, graph-
ical artists and Web/UI designers to cooperate exploiting the well defined interaction
modalities. The goal of SAGG is to go a step forward in this direction, by introducing
a further level of abstraction which is provided by interdisciplinary work of domain
experts. The SAGG provides the above-mentioned developers with rapidly deployed
mock-ups of required interfaces, possibly already working at a basic level of detail
(thus needed some fine-tunings, which are requested to the developers).

Our proposed approach is to devise a mechanism and a chain of processes (to iden-
tify a realizable architecture) that automatically generates the queries for extracting the
desired subgraph, starting from available examples, and defines the right representation
for the selected resources. The approach we are proposing has high potentials. Given a
set of knowledge items, there is a high probability that the Web contains many different
pages that have graphic widgets for showing the same knowledge items. For example,
for representing statistical data (stored in our knowledge base) it is possible to find a
Web page that contains similar data of the same format in a table then we can use this
to show our data. We can then exploit these user interfaces to infer one or the best user
interface.

By adopting Fresnel\footnote{Emmanuel Pietriga \textit{et al.}(2006)} vocabulary, it is possible to col-
lect above information in a list of pairs:

\[
< \text{Lenses, Formats} >^{2.3} \tag{5.1}
\]
5.2. Scenario

that will be used to configure the users interface and finally generate the GUI. Our method uses these examples for two reasons:

1. inducing the regularities for extracting the best graphical structure for the UI.

2. inducing the rules to automatically fill graphical structures with ontological data.

The second point is extremely importance: the representation examples on the Web do not need to have been produced by the same dataset own by the user. Instead the same example just need to contain data representing information originated from the same domain (or, at least, that share a sensible overlap with it). This information can then be searched over the real dataset owned by the user, hoping that a good percentage of the data will be recognized upon it and that the system would thus be able to induct the queries needed to extract analogous data from the dataset.

The key points of our approach are:

- Configuration Files are lively created when the user starts a learning process over data observed from a browsed example;

- The patterns of representation are determined by the users;

- The query identifying the interesting subgraphs is learnt by an automatic process;

- The information about the pattern of representation are stored and placed at user’s disposal for future sessions of navigation (or to be exported for other interested users/developers).
Chapter 5. Semi-Automatic Gui Generator (SAGG)

Figure 5.1: System Design.

5.2 Scenario

The scenario of SAGG is composed by several elements (components and relevant objects) whom interaction is shown in Figure 5.1. The components are:

- A Semantic Repository containing RDF resources;
- A GUI Generator that performed the automatic generation of user interface.

The relevant objects are:

- HTML pages the content of which is being selected by the users;
- The automatically generated UI.
5.3 Model

The RDF Repository contains one or more ontologies describing the domain of interest (the model) as well as the data provided by the user and/or retrieved during the processing of inputs. Also, data are modeled for representation purposes and describe the created pairs

$$< RDF_{resource}, representation >$$

through Fresnel Lens and Formats.

In the initial phase, if the RDF repository contains already enough data, it can immediately be used as a seed to learn new UIs from available examples retrieved from the Web, with no supervision. If the repository is empty and many similar examples are available from Web pages, it can be automatically populated by semantically annotating even very few pages (as reported in [Stephen Soderland(1999)]). A small number of annotated examples are needed when applying wrapper induction techniques to very similar pages, which are usually produced by an original pattern populated by backed data).

SAGG receives as input: HTML pages selected by the user (or highlighted parts of them), containing the formatted UI structure that the user needs to replicate, and the available RDF dataset. The SAGG analyzes the input and automatically generates a GUI mock-up with raw SPARQL queries for extracting plausible values. In section we will describe in details the architecture of SAGG.

5.3 Model

The data model behind SAGG algorithm foresees a set of elements which are identified and analyzed in standard documents. In the SAGG’s approach the DOM standard structure is adopted to make assumptions regarding the content of the page. The input of
Chapter 5. Semi-Automatic Gui Generator (SAGG)

SAGG is called 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):

\[ \text{PatternExample} = GO_0, GO_1, GO_n \]  \hspace{1cm} (5.3)

A GO is a part (presentation content) of a PatternExample exposing relevant (and mostly self-contained) information. SAGG assumes the content of GOs as independent information units i.e. information that can be analyzed separately from the rest of the page with respect to the user data. As from the adopted DOM formalism, the 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 a list of atomic information units called GE can be recognized.

\[ GO = GE_0, GE_1, GE_n \]  \hspace{1cm} (5.4)

In SAGG’s approach it is assumed that a pair of GEs contained in the same GO may be bound through a relationship. The following grammar has been defined to identify the different relations which can hold between them.

\[ (GE, +, ) \]  \hspace{1cm} (5.5)

In the grammar two types of relations have been identified:

- + 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. Note that 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 be nested on each other or belong to different branches of the DOM tree and thus not be siblings in the hierarchical organization of elements. 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 simple keyword-based search over graphs solutions), so that the algorithm can apply more constraints on the search. Thus for each GO a list of triples has been defined:

\[ < GE_x, relation, GE_y > \] (5.6)

These triples are ranked according to the following method.
First of all, we empirically assign a weight for the sibling relation (0.2) and for the dependency relation (0.4). Then to determinate the weight for each GE involved in the relation, we implement an algorithm ( see section 5.7.1 ) to check if some GEs can be mapped with some to RDF nodes of in RDF Schema the dataset. If this condition is verified, a weight equal to 0.3 is assigned to it. The final rank of a triple is the sum of the weight of its relation and weights of each of the two GEs. In the table 5.1 we show all possible combinations of values of weights for a triple.
Chapter 5. Semi-Automatic Gui Generator (SAGG)

Figure 5.2: Activity diagram for SAGG.
5.4. User Interaction

In spite of its partial automatization, the SAGG’s approach designates a centric role for users, as they provide the semantically annotated examples and, where appropriate, they validate the data retrieved from Web pages.

The interaction of users with the system consists first of all in defining what is the RDF resources that they want to represent. Then they browse the Web and search pages that have the same domain of the RDF repository.

When the user meets a graphical pattern of representation for data he is interested in, he selects it and asks the system to extract an UI widget and to induct a query for populating it with data from the available RDF resources. They do it by comparing plausibility of results from the proposed query with respect to those presented in the selected examples.

The SAGG then starts the chain of processes and propose a widget to the user, with the option of first modifying and then saving the Fresnel format, then edit the query (e.g. to add more restrictions or simply change some of its characteristics) and finally save also the Fresnel Lens.

<table>
<thead>
<tr>
<th>GEx</th>
<th>GEy</th>
<th>Relation</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>0</td>
<td>0.3</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.1: Rank Values

5.4 User Interaction
Chapter 5. Semi-Automatic Gui Generator (SAGG)

This last sequence of steps can be reiterated several times to refine the UI and change the associated query accordingly.

5.5 SAGG Web page visual analysis algorithm

In this thesis 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.

Indeed it was developed a VIPS [Maria Teresa Pazienza and Noemi Scarpato(2012)] algorithm to recognize relevant blocks (called GO (see 5.3) 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 as it makes the relations between the keywords understood. SAGG’s VIPS algorithm defines a chain of actions (see 5.7.1) to identify relations between GEs.

For a set of special kind of tags (e.g. title, table) we implemented some customized strategies, to make further assumptions regarding the nature of the relations and to identify relations of dependency between particular kinds of GEs.

To implement the 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 (more details in section 5.3).

The inputs of SAGG’s VIPS algorithm are: a Web page (or a portion of it) identified by the users in the Web; and a knowledge base indicated by the users themselves. The main assumption of SAGG’s VIPS Algorithm is that the Web page is chosen for
users because it represent 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 needed graphical representation and data that are in the same users domain.

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 the page and the visual tree has 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 them is characterized by 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 GOs have 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 first. 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) 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. Thats why in each block it is recognized a list of GEs.

In this thesis it is assumed that to understand the meaning of a GO it is useful to iden-
Chapter 5. Semi-Automatic Gui Generator (SAGG)

tify 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 we define some rules, that are related both to visual and struc-
tural features (many details in 5.7.1).
Furthermore we define a rule that allow to recognize and discard elements that are not
meaningful (e.g. comment or script that are not interesting for SAGG purpose). The
realization of this rule is achieved through the definition of a stop list of non-relevant
elements.
Same as GOs, we define a set of rules to recognize GEs. 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 we define a stop list to exclude non-relevant elements for GEs.

As shown in the figure 5.3 the idea behind SAGG Web page visual analysis algo-

rithm is to support the SAGG’s semantic search algorithm by providing as output a list
of GOs and, for each of them, the related graph of GEs that will be analyzed from the
SAGG’s semantic search algorithm. In this way the 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 states 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 we propose the SSA algorithm that is a semantic search algorithm
that exploits the SAGG’s VIPS algorithm output to improve its search process.
5.6. SAGG’S Semantic Search Algorithm

As showed in chapter 3 the retrieval of Semantic Web data is a challenging task and in the SAGG’s approach it is needed to understand the content of an example page to extract query exploited to populate generated GUIs.

This paragraph presents the SSA algorithm [Maria Teresa Pazienza and Noemi Scarpato(2012)], which is a semantic search algorithm implemented to accomplish the challenge of searching Semantic Web data (see section 5.7.2 for many details).

The SSA algorithm takes in input the output provided by the SAGG’s VIPS algorithm that is a set of GOs, composed by a set of triples in the form of:

$$< GE_1, relation, GE_2 >$$  \hspace{1cm} (5.7)
Chapter 5. Semi-Automatic Gui Generator (SAGG)

In the figure we show the interaction between SAGG’s SSA Algorithm and SAGG’s VIPS Algorithm 5.3.

The SSA algorithm for each of the previous triples identifies a related list of triples composed of:

\[
< \text{keyword}_1, \text{relation}, \text{keyword}_2 >
\]  

(5.8)

To extract keywords from textual content of both GEs in the triple, SSA uses the Chaos parser [Roberto Basili et al. (2001)]. Chaos is a modular and lexicalized syntactic and semantic parser for Italian and English languages. In the SSA approach, the syntactic parsing features are used to identify 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) into the textual content of the GEs.

The second phase of the SSA algorithm for each GO analyzes the list of previous triples (see 5.8) to map them in the RDF graphs by the following actions:

- In the RDF graph the SSA algorithm tries to map each keyword into a node;

- The relation is investigated using an approach guided by the nature of the relation identified in SAG’s VIPS algorithm (i.e. if a dependency relation is recognized, the SSA investigates first of all “InstanceOf” and “SubClassOf” and only if none of these are reify, other relations in the knowledge base are investigated);

- After the validation of triples, these are analyzed and composed to create a list of queries related to each GO.

The SSA algorithm differs from [Yuanghai Lei et al. (2006)] and other semantic search approaches mentioned in [G.Madhu et al. (2011)] because they take as input only queries
composed by list of keywords. SSA instead exploits the output of SAGG’s VIPS Algorithm that is composed by a graph of keywords. This approach makes the composition of candidates matching graph patterns (lists of triples) easier in comparison with a simple list of keywords.

The SAGG system proposes a new interaction mode between the users and the semantic search system, asking users to indicate only an example of a Web page that contains both keywords and visualization patterns instead of semantic or keywords query. The SSA algorithm implements an algorithm of query expansion guided by the definition of relationship identified by the analysis provided by the SAGG’s VIPS Algorithm [5.5]. The SSA algorithm tries to map not only the keywords, but the entire triple and this allows to identify both resources and relations.

The later sections describe in details the architecture of the SAGG system and its modules, that implements the SAGG’s approach from the analysis of the Web page, to the creation of the GUI.

5.7 Architecture

The architecture of the SAGG system is based on the visualization process [1.2] because it provides all of three steps of this process (selection, structuring and formatting). In the SAGG architecture, it is implemented the reusability and shareability of the information that are worthwhile in all Semantic Web applications. The architecture of the SAGG system is structured as a flow involving a chain of different modules, each of them providing an output that is used by one or more of the following modules.

The architecture of SAGG system is composed by four modules plus a data man-
Figure 5.4: Semi-Automatic GUI Generator Architecture.
5.7. Architecture

ager to access the user’s semantic repository (see: figure 6.3). The first three modules of the architecture are involved in achieving each of the three steps of the visualization process.

The first module of the SAGG architecture is devoted to the pattern generation and carries out, along with the Query Generator, the selection step of the visualization process. This module implements a Web page segmentation algorithm to identify graphic objects (GO) and their graphic elements (GE) (paragraph 5.5). Furthermore, the same module generates a list of triples as:

\[ < GE_i, relation, GE_y > \]  \hspace{1cm} (5.9)

these triples contain the relations between all pairs of GEs that are part of the same GO. A key aspect of this processing is that the structure of the GO is exploited and in particular the position of GEs in a recognized GO is analyzed to establish relationships between GEs and to assess the nature of these relationships.

The second module: Query Generator, implements a semantic search algorithm that takes as input a set of GOs with the associated list of triples generated by the Pattern Generator module. The Query Generator module analyses the matching graph patterns to induct the query which best approximates the data contained in the input. This process is performed for each of the GOs. The output of this module consists of a list of GOs and related queries able to populate them.

In the first two modules we implement two algorithms to identify patterns of representation and queries able to fill them, implementing the selection step of the visualization process.
Chapter 5. Semi-Automatic Gui Generator (SAGG)

process. The following modules instead are in charge of generating the user interface.

In the SAGG system, we adopt the Fresnel vocabulary (see: 2.3) to create the user interface. The Configuration File Composer module, uses both the patterns and queries provided by the previous modules to create a Configuration File modelled according to the Fresnel vocabulary.

The Pattern Generator identifies the patterns used to generate the Formats (see: 2.3.2). Instead the Query Generator provides the queries to generate Lenses (see: 2.3.1). In the SAGG’s approach, the Configuration File composer implements both the structuring and the formatting steps of the Visualization Process through the definition of both Lenses and Formats.

The GUI Composer uses the Configuration File, to automatically generate a GUI capable of showing the user’s dataset organized according to the required pattern of visualization. Moreover the produced Configuration Files are stored into a knowledge base and can be shared with other users. This module has been designed as capable to parse a Fresnel file and to create a form using the information available. The form can be exported in different Formats as HTML, JSP, XUL, form java etc. In section 5.8 we present an example of the application of the SAGG system to generate an HTML form.

The last module of the SAGG’s architecture is the Data Manager; it provides an interface to access the user dataset and is realized through the OWLArt API.

This API has been developed inside the ART Research Group at the University of Rome, Tor Vergata. OWL ART API are being adopted in all of ART’s Semantic Web tools, such as OntoLing (a tool that allows for Linguistic Enrichment of Ontologies) and Semantic Turkey

\[\text{Emmanuel Pietriga et al.}(2006)\]

\[\text{Maria Teresa Pazienza and Armando Stellato}(2006)\]

\[\text{http://art.uniroma2.it/owlart/}\]
They offer an abstraction layer over different RDF triple store technologies, so that applications can use them and easily switch among different storage technologies without adapting source code to their API. Considering previous motivations, in the SAGG system the OWLArt API has been integrated into the Data Manager module.

### 5.7.1 Pattern Generator

This section describes the rule-based VIPS algorithm upon which the Pattern Generator module is based.

In the description of the algorithm, we adopt an input that is composed by an HTML Document, but the flow of the algorithm keeps the same execution sequence for any type of analysed documents.

The first task of the algorithm is the identification of graphic objects (GO) and their graphic element (GE) in the PatternExample.

The input of the SAGG system can be an entire document or part of this (see: figure 5.1), in the first case identification of all the GOs contained in the input page is needed, while in the second one the SAGG system assumes that the portion of the document given in input contains only a GO.

To recognize a GO the following list of rules has been defined:

- An element is not a GO if it is empty;

- An element is not a GO if it corresponds to an element of the Stop List (as example for HTML document the Stop List is: comment, script, doctype, metadata, HTML, body);
Chapter 5. Semi-Automatic Gui Generator (SAGG)

- An element is not a GO if it is an inline element (this rule can be applied only in HTML document and it is defined through the adoption of the visual formatting model[^2]);

- If an element corresponds to an element of a list of known GO types, then it is a GO.

As shown in the algorithm[^7] the first step is the identification of GOs through the recursive application of the previous rules for each element contained in the input.

The second step of the Pattern Generator algorithm consists on checking, for each recognized GO, if this GO has a child that can be recognized as a GO itself.

In this case the algorithm creates a separated GO for each of this kind of child elements and a further GO for the remaining part of the root element. In our approach the overlap between GOs is not permitted and the part of the code related to identified GOs is then removed.

Algorithm 7 GO identification Algorithm

```plaintext
for i = 0 node_i do
    Previous rules to identify the GOs are applied;
    for y = 0 GO_y do
        if in a GO there are elements that are GO themselves then
            the GO is added separately to the list of GOs;
            Further generic GO is created for the remaining part of the root element.
        end if
    end for
end for
```

Once the algorithm identifies the GOs, it proceeds with the analysis of each GOs to identify the GEs and their relationships.

[^2]: http://www.w3.org/TR/CSS2/visuren.HTML
5.7. Architecture

To recognize the GEs we apply the following strategy:

- A GE is not empty;
- A GE is not an element of a stop list;
- If an element is part of a list of known GE types, then the element is a GE.

We show the steps to recognize the GEs in the following algorithm:

Algorithm 8  GE identification Algorithm

\[
\text{for } i = 0 \text{ to } n \text{ do}
\]
\[
\text{Previous rules to identify the GEs are applied;}
\]
\[
\text{for } y = 0 \text{ to } n \text{ do}
\]
\[
\text{if in a GE there are elements that are GE themselves then}
\]
\[
\text{the GE is added separately to the list of GEs;}
\]
\[
\text{Further GE is created, this GE contains all information that aren't in one of}
\]
\[
\text{the others GE; (For instance, if there is a P (called P₁) in a GO that contains}
\]
\[
\text{another P (called P₂) and some text, a GE for P₂ is created and a GE for P₁}
\]
\[
\text{is created only with the text outside P₂. If there is no information outside of}
\]
\[
\text{the nested GE related to P₂, the GE related to P₁ is not created.)}
\]
\[
\text{end if}
\]
\[
\text{end for}
\]
\[
\text{end for}
\]

Next phase of the algorithm is the generation of a list of triples of the form:

\[
< GE_i, relation, GE_y >
\]

(5.10)

5.7.1.1 Strategies

In the Pattern Generator algorithm for a set of special type of tags (e.g. title, table) we implement some customized strategies to made further assumptions regarding the nature of the relations. The key idea behind this approach is that the position of GEs in a
GO is meaningful to understand the relation between them and in particular, if the kind of GO is known (as example table), it is possible to make more precise assumptions about the GEs that are part of this, and theirs relations.

Nowadays, in the SAGG system we define only a custom strategy to analyse the tables besides the general approach, but we design the Pattern Generator module to allows the introduction of further strategies.

The table strategy considers as candidate GEs only the nodes that are: table headers, column headers, row headers and table cells.

The table strategy, also, considers only the relations between the GEs that are in the same row or in the same column of the table and discards the others.

Besides, we make assumptions regarding the nature of these relations. In case of, there is a GE$_1$ that represents a column header and some GEs that represent several table cells, and that these GEs are in the same column of the GE$_1$, the strategy identifies a dependency relation between the GEs and the GE$_1$. Same assumption is provided for a GE$_2$ that represents a row header element and the GEs that are in the same row. In this cases we add some triples as:

\[
< GE_i, \text{depends}, GE_y >
\]  

(5.11)

in the list of triples. In the algorithm we show the approach that applies the strategies if they are available\[9\]

### 5.7.2 Query Generator

We describe in this paragraph the Query Generator module. The identification of proper queries able to select relevant subgraphs that satisfy the users needs is a task that is often left to the users or to the creators of metadata (see: chapters[2] and [3].
5.7. Architecture

**Algorithm 9 GE identification Algorithm**

```plaintext
if a customized strategy is available then
    the strategy to create list of triples is applied
else
    a triple for all pairs of GE that are part of the same GO is added.
end if
```

The SAGG’s approach, instead, implements the generation of the queries automatically, the Query Generator module is responsible for the identification and the creation of these queries.

The Query Generator Module takes as input the list of the GOs and the related list of the triples of GEs provided by the Pattern Generator. The Query Generator Module exploits this input and produces as output, for each GO, a list of queries that can be used to populate the patterns identified in the Pattern Generator. To achieve the aims of this module we design the SSA algorithm (see 5.6). We show in the algorithm the flow that brings to the creation of the queries:

The Query Generator module, also, provides the analysis of the generic relations for mapping them into specific relationships into the knowledge base. We define a threshold to determine which are the triples to be analyzed. In the current implementation of the Query Generator module, this threshold is fixed to 0.4 (this means that only the triples which contain at least one seed are analyzed). The threshold can be tuned to arrange the performances of the system (e.g. if the threshold is smaller than 0.4 more triples are analyzed, this can lead to the retrieval of further correspondences with the knowledge base but also can worsen the performances generating noisy information). However, the steps shown into algorithm are the same for each value of the threshold.
Chapter 5. Semi-Automatic Gui Generator (SAGG)

Algorithm 10 Query Generator Algorithm

\[
\text{for } i = 0 \text{ GO}_i \text{ do} \\
\quad \text{we analyze the list of triples } <G_{Ei}, \text{ relation}, G_{Ey}> \\
\text{for } i = 0 \text{ triple}_i \text{ do} \\
\quad \text{The keywords are extracted from textual content of both GEs by using the Chaos parser}\ [\text{Roberto Basili et al. (2001)}]. \\
\quad \text{A list of triples like: } <\text{keyword}_{i,k}, \text{ relation}, \text{keyword}_{y,k}>\text{related on } <G_{Ei}, \text{relation}, G_{Ey}> \text{is created.} \\
\quad \text{If possible the keywords related to the GE are mapped into some nodes in the knowledge base} \\
\quad \text{if keyword can be mapped in a node then} \\
\quad \quad \text{This keyword is marked as a seed (this means that this keyword has a weight equal to 0.3)} \\
\quad \text{end if} \\
\quad \text{We assign a weight value to the } <\text{keyword}_{i,k}, \text{relation}, \text{keyword}_{y,k}>\text{triple, this value is calculated as the sum of the weight values of both keywords in the triple plus the weight value assigned to the relation (this value depends on the kind of relation see paragraph 5.3 for further details)} \\
\text{end for} \\
\text{The resulting list is arranged according to the weights} \\
\text{end for}
\]

The results of previous steps are: a list of atomic queries and a list of values related to each GE related to a GOs.

5.7.3 Configuration File Composer

The Configuration File Composer module goal’s is to create a Configuration File compliant to Fresnel vocabulary. In the Configuration File Composer module we adopt the Notation 3 syntax for RDF to create the Configuration File.

To achieve the creation of the Configuration File, the module needs to create a list of general complex queries related to GOs and to define a list of general patterns of representation. These lists are used to create Groups, Lenses and Formats according to
Algorithm 11 Query Generator Algorithm 2

0: For each triple it is checked if the rank value is greater than threshold.
0: If it is:
   if rank == 1 then
   ( this means that the relation are dependency and the keywords are both a seed )
   An “InstanceOf” relation between seeds is searched into the knowledge base.
   if the triple is into the knowledge base then
   A query like: “SELECT ?x WHERE ?x a keyword_{y,k}”
   is added to the queries list associated to GE_i
   A value keyword_{y,k} is added to the value list associated to GE_y
   The same weight of the analyzed triple (1 in this case) is associated to each of
   this value. (The values are also added because they could be used to fill the GE
   in case of not valid query are identified.)
   else if This triple isn’t into the Knowledge base then
   It is searched into the knowledge base a “subClassOf” relation between seeds.
   if This triple is into the knowledge base then
   A query like: “SELECT ?x WHERE ?x rdfs:subClassOf keyword_{y,k}” is
   added to the queries list associated to GE_i
   A value keyword_{y,k} is added to the value list associated to GE_y
   The value 1 is associated to each of them.
   else
   The knowledge base is explored until a triple is found.
   if This triple is into the knowledge base then
   The related query and value, are added to proper list.
   end if
   end if
   end if
   end if
Algorithm 12 Creation of Lenses Algorithm

\begin{algorithm}
\caption{Creation of Lenses Algorithm}
\begin{algorithmic}
\FOR{$i = 0$ to $|G_O|$}
\STATE A Group that represent GO in the Configuration File is created.
\FOR{$y = 0$ to $|G_E|$}
\STATE \textit{query list and value list of GE is analyzed}.
\STATE Query list is sorted according to the triple rank,
\STATE A candidate query list with query or queries that have the most ranking value is created,
\IF{candidate query list size $> 0$}
\STATE we analyze the candidate query list to create a complex query that unify all candidate query
\STATE A Lens is created
\STATE We assign to the \textit{Fresnel:purpose} property of Lens the value \textit{Fresnel:defaultLens}
\STATE The Lens is added to the Group related to GO Lens
\ELSE
\STATE We assign to the \textit{Fresnel:purpose} property of Lens the value \textit{Fresnel:defaultLens}
\STATE The Lens is added to the Group related to GO Lens
\ENDIF
\STATE The value list is sorted according to the triple ranking
\STATE A Candidate Value list with value or values that have the most ranking value is created
\STATE We assign to the \textit{Fresnel:purpose} property of Lens the value \textit{Fresnel:labelLens}
\STATE The Lens is added to the Group related to GO Lens
\ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}

To create Formats related to the generated Lenses the pattern of representation of GEs are analysed. If in the pattern recognized by the Pattern Generator module some CSS information have been retrieved, they are integrated into the Formats. Through the OWLArt API\footnote{http://art.uniroma2.it/owlart/} the Configuration File is stored into the knowledge base and then can be reused for other visualization task or shared with others users.
5.7.4 GUI Composer

The GUI Composer module deals with the creation of the GUI shown to the users. This GUI will be filled with the knowledge base data according to the created Configuration File.

The GUI composer module, may be used separately if a well formed Fresnel Configuration File, is submitted to it in input. The main component of the GUI Composer module, is a Fresnel parser able to parse Fresnel Configuration File and to create a GUI. In the GUI Composer module the JFresnel library\(^4\) is adopted to implement the Fresnel parser. The JFresnel is a java library able to implement the Fresnel specification for various RDF APIs, such as Jena and Sesame.

The GUI composer module is able to create different kind of GUI as HTML, JSP, XUL etc. For example in chapter\(^6\) we present the integration of SAGG in Semantic Turkey , where the users can choose between an HTML GUI or a XUL GUI. We show the strategy to create the GUIs in the algorithm\(^13\).

5.8 SAGG at works

In this section we present an example of the use of SAGG. We implement a Web service able to take in input an HTML fragment and a URI of a knowledge base providing in output an HTML page.

The input is composed by a part of a document HTML that contains a table (figure 5.5).

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

\(^4\)http://jfresnel.gforge.inria.fr/
Algorithm 13 Creation of GUI Algorithm

An empty container page is created according to the type of visualization formalism chosen by users

\[
\text{for } i = 0 \text{ to } \text{GOList.size} \text{ do }
\]

A container element is created according to the Groups and it is formatted through the application of the related Forms.

\[
\text{for } y = 0 \text{ to } \text{GEList.size} \text{ do }
\]

The SPARQL query contained into SPARQL Selector of the default Lens is executed.

\[
\text{for } k = 0 \text{ to } \text{values.size} \text{ do }
\]

A child element is created, according to the default Lens and it is formatted applying the related forms.

The child element is filled with the retrieved values.

\[
\text{end for}
\]

\[
\text{for } k = 0 \text{ to } \text{labelLensList.size} \text{ do }
\]

A child according to labelLens is created and formatted through the application of the related forms.

\[
\text{end for}
\]

\[
\text{end for}
\]

three relationship: an object property “gioca_in” (play_in) that relate the concept of “Marcatore” with the concept of “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 gave 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 strategy is applied to analyze the table and 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.
5.8. SAGG at works

We show in the figure 5.6 a fragment of the triples list that contains all the triples generated for a row of the analyzed table.

The Query Generator takes in input the previous triples list and analyzes it to generate the list of atomic queries.

We show a fragment of the results of the analysis of triples related only to the last row in the figure 5.7.

The Configuration File Composer module takes in input both the lists of simple queries and values and the pattern provided by the Pattern Generator. The Configuration File Composer exploits these inputs to create the Fresnel Configuration File (see: figure 5.8).

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 5.9.

As shown in the figure 5.9, the GUI contains data retrieved from the knowledge base.

<table>
<thead>
<tr>
<th>Gol</th>
<th>Rig.</th>
<th>Marratoriato</th>
<th>Squadra</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>6</td>
<td>Antonio Di Natale</td>
<td>Udinese</td>
</tr>
<tr>
<td>22</td>
<td>4</td>
<td>Diego Milia</td>
<td>Inter</td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td>Fabrizio Miccoli</td>
<td>Palermo</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>Giampiero Pazzini</td>
<td>Sampdoria</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>Alberto Gilardi</td>
<td>Fiorentina</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>Paulo Vitor Barreto</td>
<td>Bari</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>Marco Borriello</td>
<td>Milan</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>Francesco Totti</td>
<td>Roma</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>Mirko Vucinic</td>
<td>Roma</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>Emanuele Caligi</td>
<td>Palermo</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>Alessandro Matri</td>
<td>Cagliari</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Marco Di Vare</td>
<td>Bologna</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Samuel Eze</td>
<td>Inter</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>Stefano Lucchetti</td>
<td>Lazio (4) - Chievo (4)</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>Massimo Maccarone</td>
<td>Siena</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>Alessandro Pato</td>
<td>Milan</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>Rinaldi</td>
<td>Milan</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Stefano Peluso</td>
<td>Chievo</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Fabio Gomiero</td>
<td>Napoli</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Simone Testa</td>
<td>Udinese</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Adalberto</td>
<td>Bologna</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Matti Lappi</td>
<td>Catania</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>Cristiano Lucarelli</td>
<td>Lecce</td>
</tr>
</tbody>
</table>

Figure 5.5: HTML input Table.
Chapter 5. Semi-Automatic GUI Generator (SAGG)

Figure 5.6: Triple List

Figure 5.7: Queries and values List
5.8. SAGG at works

# .................................
# 1. Groups
# .................................
calcio:th:Valuelens fresnel:value "Group" .

calcio:th1:Valuelens fresnel:value "Chapter" .

calcio:th2:Valuelens fresnel:value "Diagram" .


calcio:td1:DefaultLens rdf:type fresnel:Lens .

Figure 5.8: Configuration File.
Chapter 5. Semi-Automatic GUI Generator (SAGG)

Figure 5.9: Generated GUI.

and it is formatted like the input page.

In conclusion the Configuration File is saved in the Knowledge base alongside with the domain ontology.

5.9 Conclusion

The Semi-Automatic GUI Generator (SAGG) is a Fresnel-Based RDF Browser. The basic idea behind SAGG is to exploit the Web 2.0 which contains many examples of visualization methods that can be reused by SAGG to show RDF data in an appealing manner.

As mentioned in the previous chapters many methods and theories about visualization have been provided in the last years.

In the SAGG system the traditional approach that puts users in the center of the design of visualization systems has been followed.

One of the peculiarities of SAGG is the semi-automatic creation of the Configuration File written in Fresnel Formats. The SAGG system uses this Configuration File to generate a Widgets-based form to visualize users data.

The SAGG system tries to identify users needs analyzing the example of representation selected by users in the traditional Web. In this example, users recognize both
5.9. Conclusion

data that are in the same context and also the pattern of visualization they want to apply. The patterns of visualization are used by SAGG to understand users needs both in terms of content and in terms of visualization approaches.

The SAGG system design has been implemented through the analysis of all steps of visualization process, troubles and open challenges related to this process have been identified and finally a custom solution for each one has been proposed.

The first step of the visualization process is the selection of data, one of the main challenges of the traditional approaches is the individuation of the subsection of data that fulfill the users information needs. Many of the approaches on which Fresnel-based RDF browser are based, ask users to indicate the Lenses for their data. Instead, in SAGG both the graphical structures and the data in the examples page are used to identify queries to select data. In the SAGG’s approach, the position of data in the graphical structures is considered meaningful to determine relations between data.

The second step of the visualization process organizes the Lenses created in first step. In SAGG, this is automatically done by the system through visual analysis (see: chapter 4) of the example page to determine the order and the priorities between Lenses.

Finally the formatting step is provided by SAGG using the graphic styles of examples page to create the Formats. The SAGG system uses the css styling instructions that are present into the pages of example to create the Formats to apply to the data from selected Lenses.

The SAGG’s approach can be implemented and integrated in very different scenarios:

- Semantic Enhanced Web Browsers,
Chapter 5. Semi-Automatic Gui Generator (SAGG)

- RDF Browsers,
- Ontology Editors,
- Knowledge Management System,
- Annotation Tools.

In this thesis ST-SAGG an example of the integration between SAGG and a Knowledge Management System, has been implemented. The ST-SAGG is an extension of the Semantic Turkey system. Further details of the ST-SAGG will be provided in the following chapter.
In this chapter we describe the ST-SAGG extension, that implements the integration of SAGG system into the Semantic Turkey [Maria Teresa Pazienza et al. (2011)] system. In the section 6.1 we describe Semantic Turkey, Semantic Turkey is a complete suite for ontology browsing, managing and annotation. In the section 6.3 we provide an analysis of Semantic Turkey visualization features. Finally in the section 6.4 we present the ST-SAGG.

### 6.1 Semantic Turkey

Semantic Turkey is a Knowledge Management and Acquisition system developed by the Artificial Intelligence Group of the University of Rome, Tor Vergata. Semantic Turkey (ST, from now on) has been developed as a Web browser extension (currently implemented for the popular Web Browser Mozilla Firefox\[^1\]) for Semantic Bookmarking.

Semantic Bookmarking is the process of eliciting information from (Web) documents, to acquire new knowledge and represent it through knowledge representation standards, while keeping reference to the original information sources. Semantic Bookmarks differ from the traditional bookmarking tools because they abandon the purely partitive

\[^1\]http://www.mozilla.com/en-US/firefox/
Chapter 6. SAGG in Semantic Turkey

Figure 6.1: Semantic Turkey.
semantics of traditional links&folders bookmarking, and promote a new paradigm, aiming at “a clear separation between(acquired) knowledge data(the WHAT) and their associated information sources(the WHERE)”.

In fact, the user is able to select portions of text from the Web pages accessed from the browser, and to annotate them in a (user defined) ontology. A clear separation is maintained between ontological resources created through annotation, and the annotations themselves. This allows users to easily organize their knowledge (by establishing relationships between ontology objects, categorizing them, better defining them through attributes etc...), while keeping multiple bookmarks in a separated space, pointing to ontology resources and carrying all information related to the taken annotations (such as the page where the annotation has been taken, its title, the text which referred to the created/referenced ontology resource etc...).

Easy-to-perform drag-and-drop operations were thought to optimize user interaction, by concentrating the different actions completing the creation of both the ontological resources and their related annotations in a few mouse clicks.

ST was evolved in a complete Knowledge Management and Acquisition System based on Semantic Web technologies, by introducing full support for ontology editing and by improving functionalities for annotation&creation. Through the realization of the Range Annotator extension, Semantic Turkey has been transformed into a real Semantic Annotation System. Range Annotator extension implements an annotation mechanism that produce Range Annotations instead of the standard annotation of Semantic Turkey.

In ST a complete set of functionalities to import and export ontologies, and to manage prefix has been proposed. Users can import one or more ontologies and work
Chapter 6. SAGG in Semantic Turkey

with all of them during its work session.

Furthermore in ST a project manager feature has been introduced to allow users to create different projects that are characterized by: name, type (OWL, SKOS, SKOS-XL), URI and triple store (Allegro Graph or Sesame 2).

Another important functionality of ST is the possibility to interrogate knowledge base through SPARQL query language, that is a key feature that makes ST able to propose to users a complete suite of functionalities to create, access, manage and interrogate ontologies.

ST explores a new dimension which has no predecessor in the field of Ontology Development or Semantic Annotation, and is unique to the process of building new knowledge while exploring the Web to acquire it. ST new objective was thus to reduce the impedance mismatch between domain experts and knowledge investigators on one side, and knowledge engineers on the other, by providing them with a unifying platform for acquiring, building up, reorganizing and refining knowledge.

6.2 Semantic Turkey Extensions

ST features an extension mechanism based on a proper combination of the Mozilla extension framework (which is used to extend the user interface, drive user interaction, add/modify application functionalities and provides javascript API for the whole set of Mozilla desktop utilities) and the OSGi java extension framework which provides extension capabilities for the service and data layers of the architecture. OSGi compliance is obtained through the OSGi implementation developed by the Apache Software Foundation, called Felix. Three main extension points have been introduced: a
6.2. Semantic Turkey Extensions

Service Extension, an Ontology Manager Extension and a Data Extension, to provide respectively: new functionalities and support for other data management technologies and for introducing new application ontologies.

Both the Java business logic layer and the Javascript layer for interaction with the browser provide API[^1] for accessing/manipulating RDF data as well as for interacting with the core system and the browser hosting this application. Target users of this integrated development framework range from developers of Web browser extensions willing to add RDF-based functionalities without rewriting the whole infrastructure, to developers of knowledge acquisition tools, which get all the basic ontology management functionalities and the possibility to interact with Web content through Web browser (and its specific development environment). In this sense, ST extensions may range from simple functionalities added to the basic tool, to completely new applications hosted on the Web browser, which just rely on the underlying infrastructure for knowledge management. A success story in this sense is offered by STIA [Maria Teresa Pazienza and Noemi Scarpato(2009)], which provides a totally new annotation environment for comparing Web documents in the jurisprudence domain and for matching concepts from different laws, completely hiding underlying ontological details. Further examples of extensions of ST are MASKKOT [Armando Stellato et al.(2010)] and ST-OL [Fallucchi et al.(2009)].

[^1]: Interaction with the business logic of the system is provided by direct Semantic Turkey API, access to RDF is provided by OWL ART API (http://art.uniroma2.it/owlart/); the hosting browser is accessible through Mozilla Javascript language while Javascript API (http://semanticturkey.uniroma2.it/documentation/jsdoc) allow for access to the service layer functionalities from the presentation layer.
6.3 Semantic Turkey ontologies visualization feature

ST proposes three visualization modes to browsing ontologies: the Ontology panel, the Resource panel, and the Graph View.

The Ontology Panel Tree is located at the left of the screen in a toolbar. It is composed by the class and the property panel. The class panel is divided in two parts, the upper panel includes class tree and the bottom panel contains the instance list related to the selected class in the class panel. (i.e. when in the tree one class is selected, in the bottom panel it is shown the list of instances of the selected class). The Property panel instead contains the property tree.

When an user makes a double click on a resources in the Ontology panel, the related Resource panel is shown containing all the characteristics of the selected resource. This panel is built differently depending on the type of resources selected (class, instance, property etc).

The Graph View is implemented through a java applet that is loaded on a new tab of the browser. In this tab the graph view of the ontology is displayed, allowing the user to navigate its content.

The nodes of the graph can be displayed in different ways, according to the nature of the ontological entity: classes, properties or individuals. By dragging the mouse pointer on a node that represents an individual, it is possible to popup a window, which contains the URLs of the pages where that instance has been annotated.

Semantic Turkey provides users with three traditional visualization modes that present the common problems of the knowledge visualization system that have been discussed in chapter 2. To solve this problems we have implemented the ST-SAGG
6.3. Semantic Turkey ontologies visualization feature

Figure 6.2: Semantic Turkey Graph View.
extension which integrate SAGG systems in ST. As mentioned in chapter SAGG proposes an innovative approach for the ontologies visualization; we will describe the integration of this approach in ST in the next section.

6.4 ST-SAGG

The integration between SAGG and Semantic Turkey improves both the Semantic Turkey and SAGG systems.

On one hand the users of Semantic Turkey can benefit from the visualization peculiarities of the SAGG system and from the creation of the Configuration File for their own RDF resources.

On the other hand the users of the SAGG system can take advantage from the Semantic Turkey ontology editing features to store, modify and share visualization information.

There are many motivations that expedite the integration of Sematic Turkey and SAGG:

- In SAGG knowledge base, interaction is build on OWLART API as described in chapter also Semantic Turkey manages its own knowledge base trough the OWLART API then it is clear that the integration of the two systems allows the users to manage the same knowledge base with the same API;

- Semantic Turkey is a Firefox extension and provides a drag & drop feature that allow users to select and copy information from the Web to the system. Considering that the SAGG’s users need to browse the Web to identify relevant example of representation, the integration between the two systems allows users to exploit drag & drop feature with the aim to select and submit the example pages;
6.4. ST-SAGG

- Semantic Turkey provides a set of ontology management features that can be used to manage both the ontology and its visualization information that is represented in SAGG using Fresnel vocabularies;

- Semantic Turkey allows users to import the ontologies. Considering that in SAGG the Configuration Files are written using a RDF display vocabulary (i.e. Fresnel), the integration of the two systems allows the import of the existing Configuration Files;

- Semantic Turkey allow users to export ontologies. Basing on this in SAGG the Configuration Files are written in Fresnel (therefore these can be considered as visualization ontologies), then it is possible to export them to allow reusing and sharing;

- SAGG can improve the Semantic Turkey visualization features that present the common problems of knowledge visualization systems as discussed in chapter 2.

For these reasons a Semantic Turkey Extension called ST-SAGG has been developed.

We show the ST-SAGG architecture in the figure 6.3.

We carried out the interaction between Semantic Turkey and the ST-SAGG exploiting the Semantic Turkey extensions mechanism.

ST-SAGG is composed by two main components: the ST-SAGG UI extension and the ST-SAGG service extension.

The ST-SAGG UI extension enriches the interface of Semantic Turkey to introduce the elements that allow users the interaction between ST-SAGG extension.

The first element introducing by ST-SAGG is the “create GUI” option. This option
is added in the Metadata Panel of Semantic Turkey in the context panel related to the ontologies list. It can be used to submit an example page that must be analyzed by SAGG. When the users select the “create GUI” option, it is opened a panel that allows to indicate the example page. To submit example page, users can insert an URI that indicate the address of the example page or can insert a part of the code contained in the example using the drag&drop from the example page to ST-SAGG panel. When the SAGG process is ended, a new tab in Firefox with create GUI is opened.

We also add in the context menu related to the ontologies list, a “manage GUI” option.
This option can be selected to manage the existing Configuration File. When the “man-
age GUI” option is selected, a panel containing the list of existing Configuration Files is shown and the users can select one of this files to visualize a generated GUI in a new tab.

The ST-SAGG service component implements the request mechanism and replies to each request submitted by the users (as example manage GUI or generate GUI). For example when the ST-SAGG service component receives the “generate GUI” request, the ST-SAGG starts the SAGG process analyzing example and producing GUI.

### 6.4.1 User Interaction

Users interaction with ST-SAGG can be split in two use case: create GUI and manage GUI.

The “create GUI” use case consists of the following sequence of steps:

- User defines which is the RDF resources he wants to represent;
- User creates a new Semantic Turkey project with the selected RDF resources;
- User browses the Web and searches for a web page in the same domain of his RDF repository;
- When the user meets a graphical pattern of representation in which he is interested, he selects it and asks to the SAGG system to generate a new GUI by selecting the “create GUI” option in ST-SAGG UI;
- The ST-SAGG service invokes the SAGG and it starts its process to provide a GUI to the user through the Fresnel Configuration File automatic generation;
Chapter 6. SAGG in Semantic Turkey

Figure 6.4: Activity diagram for ST-SAGG.

- The Fresnel Configuration File is saved into the knowledge base.

We show the activity diagram of create GUI use case in the figure 6.4.
The “manage GUI” use case is composed by the following steps:

- User selects the “manage GUI” option.

- ST-SAGG shows the “Manager Panel” filled with the list of the existing Configuration File;

- User selects the Configuration File that he wants to edit and chooses the editing operation that he wants to carry out on the Configuration File;
6.5 Conclusions

- If user chooses the “show GUI” option a new tab with a generated GUI is opened in the browser;

- If user chooses “change Configuration File” option, a new panel with the list of Configuration File elements (i.e. Lenses, Group and Formats) is shown and the user can select one of these elements and modify it.

6.5 Conclusions

As mentioned in chapter 5, the SAGG’s approach can be implemented and integrated in very different kind of scenarios. We carry out an implementation of this integrated approach in the ST-SAGG.

As mentioned above the ST-SAGG has been implemented as an extension of the Semantic Turkey knowledge management system. This integration is useful for many different reasons.

Firstly, we adopt in the implementation of SAGG the OWLART API that are also adopted in Semantic Turkey; secondly Semantic Turkey is an extension of Mozilla Firefox browser, which facilitate the user interaction with SAGG (in SAGG the retrieval of a Web page example is required, and then its integration in a browser can facilitate the retrieval).

The integration is also required because Semantic Turkey provides a complete suite for the ontology editing that can be used in the managing of the Configuration Files and allows to import external and export internal knowledge resources. For this reasons the SAGG’s users can share their Configuration Files.

While integration of SAGG in the Semantic Turkey is enjoyable for many reasons, this is not the only option, as SAGG can be used as a stand-alone system or can be
integrated in different kind of systems as annotation tools and ontology editors.
Part IV

Conclusions
Conclusions and future Works

In this thesis we present an innovative approach to generate visualization forms able to show linked data in an appealing manner. To the best of our knowledge, this approach represents an innovative contribution in the knowledge-based visualization research field.

In chapter 5 we describe the data model behind our system (SAGG) and the architecture which implements it. Each module of the architecture deals with one different aspect of the visualization process and all of them provide the analysis of pattern of representation and the generation of structured queries exploiting the selected graphical objects.

The SAGG also implements the generation of Configuration File according to Fresnel W3C specification and finally it carries out the generation of customized GUI to show data.

Differently from other approaches in literature, our method can interact with different domains as the SAGG asks users to indicate their own knowledge base and generates 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 chapter 5, for this reason, we presented the ST-SAGG which
Chapter 7. Conclusions and future Works

is an extension of Semantic Turkey to let users visually select the pages or a part of them and visually follow all the steps up to the production of the GUI (and its implementation to XUL, being Semantic Turkey hosted on the Firefox Web Browser) and its use inside the same editing and visualization tool.

In this thesis, to generate GUI in semi-automatic way, we propose a novel VIPs algorithm which is the SAGG’s VIPs algorithm. This introduces a further level of segmentation with respect to the traditional VIPS algorithms, indeed not only the visual object (called GO in this approach) but also theirs 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 we implemented an innovative semantic search algorithm: the SSA algorithm. The SSA algorithm exploits the output of SAGG’s VIPs algorithm to perform an optimized search process.

In addition to the visualization feature, our approach can be exploited to facilitate the ontology population. Indeed as mentioned in chapter SAGG identifies not only the candidate queries but also the values that could be used to fill the GEs. If these are not in the knowledge base they can be added to the generated GUI in any case and can be proposed to the users. If the values are validated, they are stored in the knowledge base.

In conclusion we can affirm that the SAGG system has reached the goal of providing a RDF user-friendly browser with a very appealing graphical interface. Furthermore the SAGG system 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 we implemented
two innovative algorithms (the SAGG’s VIPS algorithm and the SSA algorithm).

7.1 Future Works

A future research direction for SAGG systems is the exploring of the possibility of combining several Configuration 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 process resulting in more complex Fresnel configurations, 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 for reusable and shareable libraries of active UIs (i.e., bringing with them the information on how to populate them starting from the available data), which should be easily searched (according to different perspectives: what they show and how they do 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 explore the possibility to expand our model to include the formalization of relations between GOs with the aim to extend the VIPS algorithm (SAGG VIPS) and consequently provide a more informative input to the semantic search algorithm (SSA). The exploration of this kind of relations is based on the idea, as mentioned above, that not only the significance of the position of the GEs in a GO, but also for the GOs their 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 strengthen the identification of the dependence relations between
Chapter 7. Conclusions and future Works

GEs in different kind of GOs (not only table but also paragraph etc..).
Bibliography


[Ben Adida and Mark Birbeck(2008)] Ben Adida and Mark Birbeck (2008). RDFa Primer - Bridging the Human and Data Webs. (Cited on pages 6, 7, 46, and 67.)


BIBLIOGRAPHY


BIBLIOGRAPHY

(Cited on page 20.)

International Conference on Publication, pages 1374–1379. (Cited on page 61.)

(Cited on pages 50 and 80.)