Semantic Navigation on the Web with SWGET

Valeria Fionda[1], Claudio Gutierrez[2], Giuseppe Pirró[1]

[1] KRDB, Free University of Bozen-Bolzano, Bolzano, Italy
[2] DCC, Universidad de Chile, Santiago, Chile

Abstract. Semantic navigation in the Web of Data is crucial to exploit the power of thousand of RDF data sources today available. We present SWGET, a tool that enables to perform selective navigation of distributed semantic data sources, triggering of actions over data encountered during the navigation, retrieval of data and extraction of relevant Web fragments. At the core of SWGET there is a powerful navigational language called NautiLOD with a concise syntax and a formal semantics. SWGET can be exploited to write declarative specifications of information on the Web in the form of scripts that can be shared, mixed, and reused. We describe the architecture of the SWGET tool and present both a standalone version and an online portal where users can create their intelligent agents, launch them and be notified when results are ready. Besides, SWGET also provide an appealing visualization tool to explore and make sense of results.

Key words: Semantic Navigation, Scripts, Linked Open Data

1 Introduction

We are assisting to a renewed interest in the graph nature of Web data. On one hand, initiatives such as the Google Knowledge Graph and Facebook Open Graph, although underlying the importance of semantic relations between data items, adopt closed architectures with idiosyncratic data models and very basic query languages. For instance, it is not possible to perform complex information requests involving navigation of (distributed) data sources. On the other hand, huge amounts of open data are shared on the Web by using standards such as the Resource Description Framework (RDF) for publishing and interlinking data and query languages such as SPARQL to query these data. This revolution is turning the classical Web, focused on hypertext documents and syntactic links among them, into a Web of Data. In this new setting, Uniform Resource Identifiers (URIs) are used not only to identify Web documents and digital content, but also new kinds of resources such as real world objects (e.g., people, places, football teams) and abstract concepts (e.g., sport, philosophy, geography). Descriptions (or representations) for these resources can be obtained, in the same spirit of traditional documents, by dereferencing their associated URIs via the HTTP protocol. Semantic links and descriptions are expressed by using a common data format, that is, RDF. Fig. 1 shows the parallel between the *traditional* Web and the Web of Data. In particular, it reports an excerpt of Wikipedia and its counterpart in the Web of Data, DBPedia. While the former is based on documents and their hyper-links, the latter is founded on resources and semantic descriptions. In the right part of Fig. 1, each dashed-circle represents a data source, identified by a URI, containing the RDF description of the resource. For instance, in the data source associated to the singer Robert Johnson, there is an RDF triple stating that he died in the city of Greenwood. Note the semantic link between the corresponding resources expressed via the property `dbp-onto:deathPlace` defined in the DBPedia ontology.

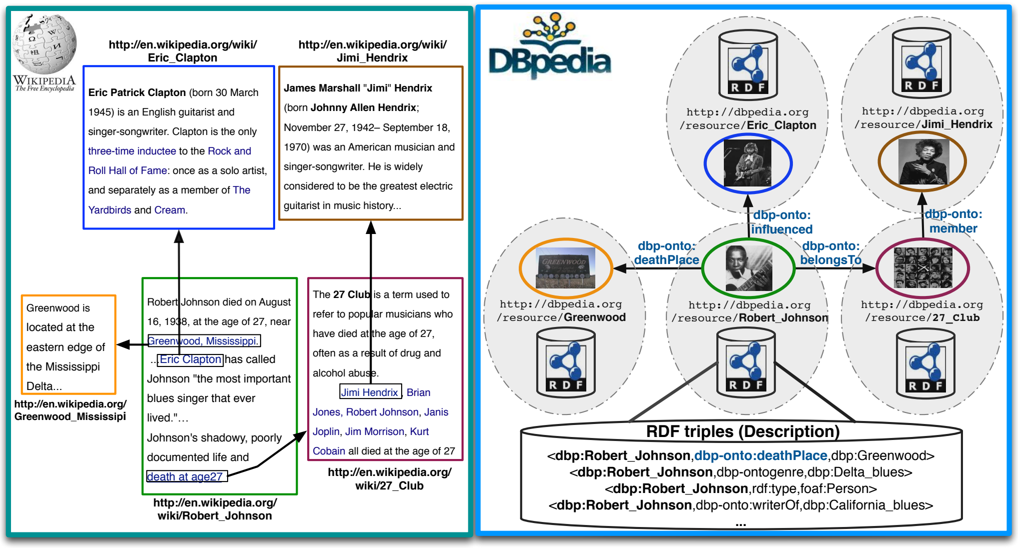

http://en.wikipedia.org/wiki/Eric_Clapton
Eric Patrick Clapton (born 30 March 1945) is an English guitarist and singer-songwriter. Clapton is the only three-time inductee to the Rock and Roll Hall of Fame: once as a solo artist, and separately as a member of The Yardbirds and Cream.
http://en.wikipedia.org/wiki/Jimi_Hendrix
James Marshall "Jimi" Hendrix (born Johnny Allen Hendrix; November 27, 1942– September 18, 1970) was an American musician and singer-songwriter. He is widely considered to be the greatest electric guitarist in music history...
Greenwood is located at the eastern edge of the Mississippi Delta...
http://en.wikipedia.org/Greenwood_Mississipi
Robert Johnson died on August 16, 1938, at the age of 27, near Greenwood, Mississippi. ...Eric Clapton has called Johnson "the most important blues singer that ever lived."... Johnson's shadowy, poorly documented life and death at age27
http://en.wikipedia.org/wiki/Robert_Johnson
The 27 Club is a term used to refer to popular musicians who have died at the age of 27, often as a result of drug and alcohol abuse. Jimi Hendrix, Brian Jones, Robert Johnson, Janis Joplin, Jim Morrison, Kurt Cobain all died at the age of 27
http://en.wikipedia.org/wiki/27_Club
DBpedia
http://dbpedia.org/resource/Eric_Clapton
http://dbpedia.org/resource/Jimi_Hendrix
dbp-onto: influenced
dbp-onto: member
dbp-onto: deathPlace
dbp-onto: belongsTo
http://dbpedia.org/resource/Greenwood
http://dbpedia.org/resource/Robert_Johnson
http://dbpedia.org/resource/27_Club
RDF
RDF triples (Description)
<dbp:Robert_Johnson,dbp-onto:deathPlace,dbp:Greenwood>
<dbp:Robert_Johnson,dbp-ontogenre,dbp:Delta_blues>
<dbp:Robert_Johnson,rdf:type,foaf:Person>
<dbp:Robert_Johnson,dbp-onto:writerOf,dbp:California_blues>
...

Fig. 1. Web of Documents versus Web of Data.

1.1 Why Semantic Navigation on the Web?

Initiatives such as the Google Knowledge Graph underline the importance of semantic relations between data items. For instance, by submitting the keyword *Eric Clapton*, one gets some structured information such as his birth date, his albums and so forth. However, the query mechanism is still keyword based and lacks any support for graph navigation; it is not possible to specify patterns to select relevant information. The Knowledge Graph only provides a set of related *nodes* (e.g., the album Crossroads) from where to manually continue the navigation. We claim that in order to harness the full potential of graph-like data available in the Web, it is crucial to have navigational languages that enable to perform automatic navigation via a declaratively specification of the parts of the Web of interest. Navigational languages enable finding pairs of nodes connected by a sequence of edge labels matching some pattern (or navigational expression) expressed via regular expressions over the alphabet of edge labels. In a Web context, since the structure of the graph is unknown, a seed node where the navigation starts is provided[3]. The Web of Data and its inherent features, i.e., standard RDF data sources and usage of well-established technologies (e.g., the HTTP protocol), is an attractive environment where to apply navigation at large scale. RDF properties connecting data items and SPARQL to query RDF data sources enable to rise navigation to the level of semantic navigation. Consider the scenario depicted in Fig. 1. Starting from Robert Johnson in `dbpedia.org`, navigation enables to discover musicians that he influenced such as Eric Clapton or members of the 27 Club. This kind of navigation on a single data source (i.e., DBPedia) can be performed by using SPARQL 1.1. despite its limitations as the lack of branching in property paths. The goal of the SWGET system is to "give the power" to Web users. SWGET enables to write script that declaratively specify and enable to semantically navigate autonomous datasources in the Web. A typical scenario where SWGET operates is depicted in Fig.3.

[3] The seed node in the Google Knowledge Graph was the keyword Eric Clapton (more precisely, its internal identifier in the Knowledge Graph).

2 The swget system

The main objective of the swget tool is to enables users to write scripts containing navigational expressions to be evaluated over the whole Web. swget implements the formal navigational language NautiLOD described in our previous work[4] and is available in three flavours: i) a command line tool; ii) a standalone GUI; iii) a Web application where users can create scripts, submit them and be notified when results are ready. swget has been implemented in Java by exploiting technologies such as the HTTP protocol to retrieve data directly from the source, JavaCC to deal with the features of regular languages, Prefuse to visualize RDF graphs, Adobe Flex to build the Web application. The tool standalone is downloadable at `http://swget.wordpress.com` while the Web application is accessible at `http://swget.inf.unibz.it`.

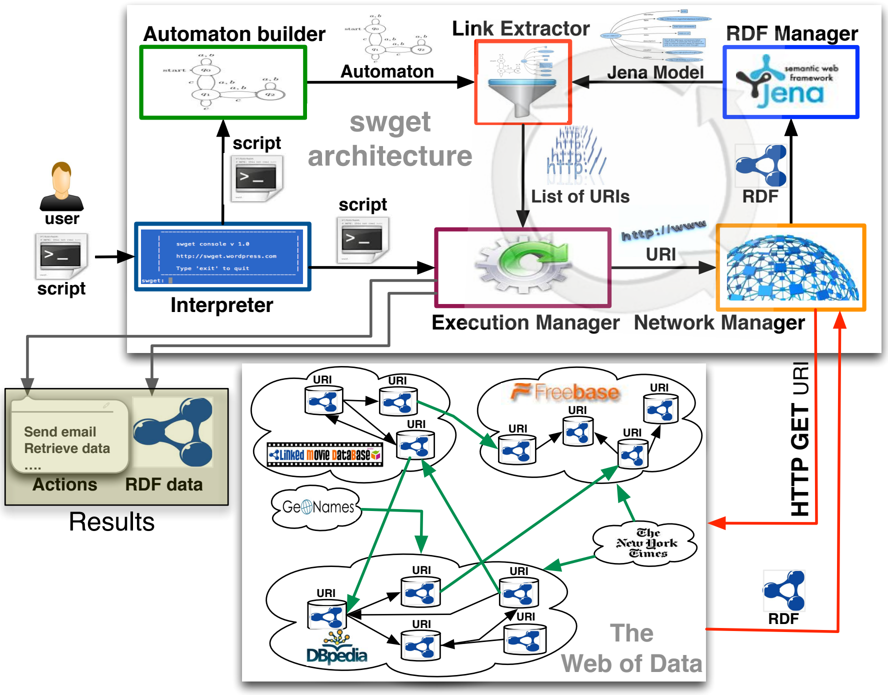


Fig. 2. The swget architecture.

2.1 High level architecture

The high level architecture of swget is reported in Fig. 2. The user submits to the system a swget script defined according to the specification described in Section 2.2.The *Interpreter* receives the input, checks the syntax and passes it to the *Automaton Builder*. From the script, the *Automaton builder*, generates the automaton associated to the navigational expression, which will be used to drive the execution of the script on the Web. The execution Manager, controls the flow of the execution and passes to the *Network Manager* the URIs to be dereferenced. This module performs the dereferencing of URIs via HTTP

[4] Fionda V., Gutierrez C., Pirró G.: Semantic Navigation on the Web of Data: Specification of Routes, Web Fragments and Actions. In *Proc. of WWW*, pp. 281-290, (2012).

GET calls and obtains set of RDF triples, which are converted into Jena models by the *RDF Manager*. The *Link Extractor* module takes in input the automaton and the model and selects a subset of outgoing links (to be expanded at the next step of the navigation) according to the current state of the automaton. The set is given to the *Execution Manager*, which starts over the cycle. The execution will end either when some navigational parameter imposes it (e.g., a threshold on the network traffic has been reached) or when there are no more URIs to be dereferenced.

2.2 SWGET syntax

In this section we describe the abstract syntax through which scripts can be defined. SWGET scripts are written in RDF and contain a set of triples along with navigational expressions in the NautiLOD language, the syntax of which is reported below.

$$
\begin{array}{rl}
\texttt{path} ::= & \texttt{pred} \mid \texttt{pred}^{-1} \mid \texttt{action} \mid \texttt{path/path} \\
 & \mid (\texttt{path})? \mid (\texttt{path})* \mid (\texttt{path}|\texttt{path}) \mid \texttt{path}[\texttt{test}] \\
\texttt{pred} ::= & <\textbf{RDF predicate}> \mid <\_> \\
\texttt{test} ::= & \textbf{ASK-SPARQL query} \\
\texttt{action} ::= & \textbf{procedure[Select-SPARQL query]}
\end{array}
$$

NautiLOD provides a mechanism to declaratively: (i) define *navigational expressions*; (ii) allow *semantic control* over the navigation via test queries; (iii) *retrieve data* by performing actions as side-effects along the navigational path. The navigational core of the language is based on regular path expressions, pretty much like Web query languages and XPath. The semantic control is done via existential tests using ASK-SPARQL queries. This mechanism allows to redirect the navigation based on the information present at each node of the navigation path. Finally, the language allows to command actions during the navigation according to decisions based on the original specification and the local information found. Now we are ready to introduce SWGET scripts for which an ontology, supporting their semantic specification has been defined.

Definition 1 *(*SWGET script*). A* SWGET *script $\mathcal{S}$ is a tuple of the form $\langle n, G, s, e\rangle$, where n is the URI, which defines the name of the script, G is an RDF graph, s is the seed URI where the navigation starts and e is a* NautiLOD *expression.*

To explain how SWGET scripts look like, consider the following request to be evaluated over the excerpt of Web depicted in Fig.3.

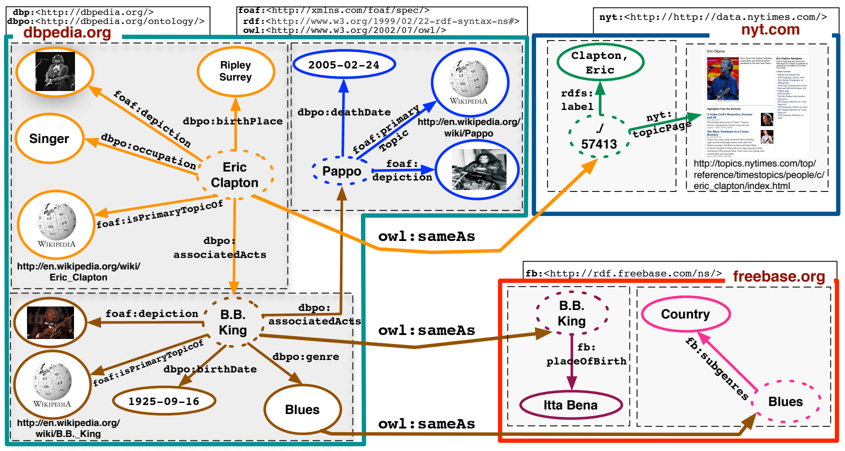
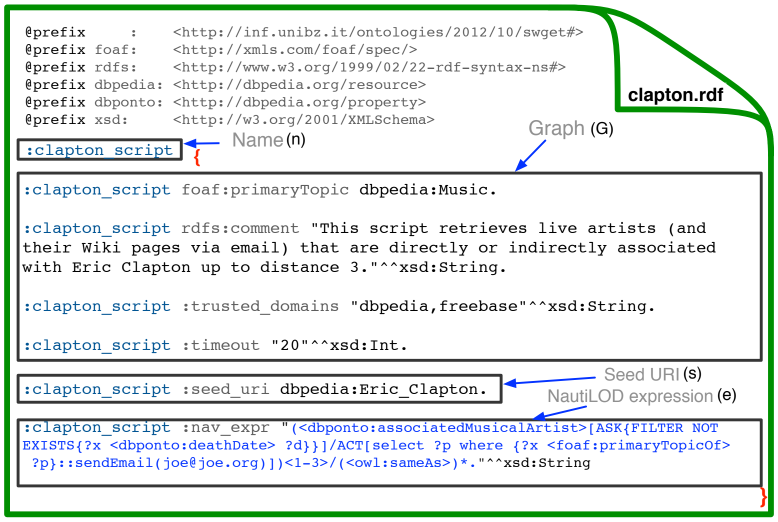


Fig. 3. An excerpt of the Web of Data with information from different datasources.

Example. *Joe is a fan of Eric Clapton and wants to discover artists (and their aliases) (in)directly associated with Clapton up to distance 3. In particular, he is interested in chains of artists that are still alive and wants to receive via email their Wiki pages.*

In order to fulfil this request via SWGET Joe writes the script reported in Fig. 4. Let's explain how it has been built. The first thing to do is to create a new script and give it a name (i.e., `clapton.rdf`). Then a graph G can be defined with triples stating, for instance, the topic of the script (i.e., Music), a comment in natural language to facilitate its reuse and so forth. Besides, in G some parameters to bound the portion of the network visited can be also defined. Here, it is stated (property `:trusted_domains`) that only information from `dbpedia.org` and `freebase.org` should be trusted and further processed. Also a timeout has been set (property `:timeout`). Further options are described in the Web site.



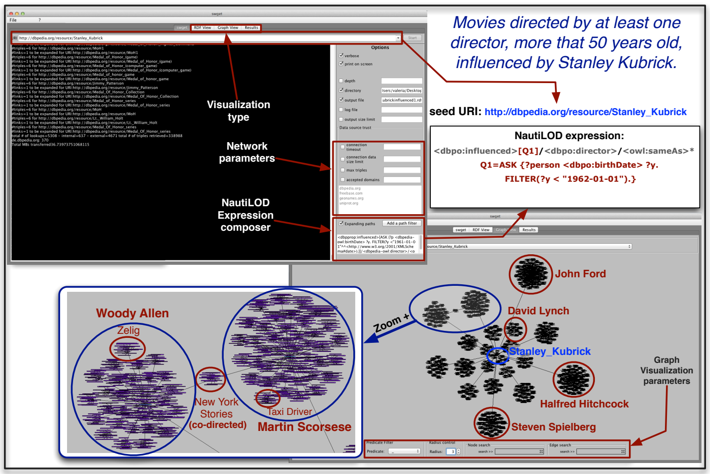
```
@prefix     :     <http://inf.unibz.it/ontologies/2012/10/swget#>
@prefix foaf:     <http://xmls.com/foaf/spec/>
@prefix rdfs:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix dbpedia: <http://dbpedia.org/resource>
@prefix dbponto: <http://dbpedia.org/property>
@prefix xsd:      <http://w3.org/2001/XMLSchema>
:clapton_script
{
:clapton_script foaf:primaryTopic dbpedia:Music.

:clapton_script rdfs:comment "This script retrieves live artists (and
their Wiki pages via email) that are directly or indirectly associated
with Eric Clapton up to distance 3."^^xsd:String.

:clapton_script :trusted_domains "dbpedia,freebase"^^xsd:String.

:clapton_script :timeout "20"^^xsd:Int.

:clapton_script :seed_uri dbpedia:Eric_Clapton.

:clapton_script :nav_expr "(<dbponto:associatedMusicalArtist>[ASK{FILTER NOT
EXISTS{?x <dbponto:deathDate> ?d}}]/ACT[select ?p where {?x <foaf:primaryTopicOf>
 ?p}::sendEmail(joe@joe.org)])<1-3>/(<owl:sameAs>)*."^^xsd:String
}
```

Fig. 4. A SWGET script.

The next step is to specify (property `:seed_uri`) the seed URI where the navigation starts. In this example the navigation starts from the URI associated to Eric Clapton in DBpedia. The last step is to define (property `:nav_expr`) the navigational expression in NAUTILOD. The property `associatedMusicalArtist` is exploited to discover chain of associated artists (see Fig. 3). However, as Joe is interested only in chains including artists that are still alive, the ASK query is used to select only those artists (it keeps artists for which the property `dbpo:deathDate` does not exist). From those artists (B.B. King in this case), the system triggers an action (described by the `ACT []` block) that selects on the current datasource (i.e., triples about B.B. King in Fig. 3) the Wikipage and send it via email. Actions can be considered as a side-effect that do not interfer with the navigation, which continues toward the datasource in `freebase.org` associated to B.B. King reached via `owl:sameAs`. Since form here, it is not possible to further expand `owl:sameAs` link the navigation ends. Note that from B.B. it is also possible to reach the datasource associated to Pappo in DBPedia. However, since Pappo did not pass the test define in the ASK query (he is not alive) this navigation branch ends.

3 Use case scenarios

Scenario 1 (standalone GUI): Joe is fond of cinema and is a fan of Stanley Kubrick. He maintains a Web page with information about Kubrick collected from different Web sites such as Wikipedia where he found information about Kubrick's life and basic information about his movies, and IMDB where he found more detailed information about Kubrick's movies. Joe wonders whether other directors, which have been influenced by Kubrick have directed interesting movies worth to be mentioned in his Web page. The idea seems appealing but: *what if Joe tomorrow is interested in some other directors?* He realizes that the burden of manually retrieving information (from different sources) is too much; besides, on a regular basis he has to "manually" look for relevant information to be added in his Web page. Joe has a thought: *it would be nice to have an intelligent tool that automatically "navigates" the Web on my behalf and find relevant information.* He realizes that DBPedia, LinkedMDB and Freebase maintain information very akin to that he manually collects. The last tile of the puzzle remains: finding the tool! Joe has been told about SWGET, which may help him. He visits the website and has a quick look at the syntax of NautiLOD. He downloads the tools and after launching it he gets the GUI shown in Fig. 5 (in the figure for sake of space the main view and the graph view are represented together). Joe writes an SWGET script to fulfil his information needs. The seed URI is that of Kubrick in DBPedia while the NautiLOD expression is reported in the top-left part of Fig. 5. Joe decided to consider movies directed by some director directly influenced by Kubrick with the constraint that these directors have to be more than 50 years old. Joe launches the scripts and goes to the pub with his friends: the tool in the meantime will do the job. When he comes back, everything is done. Joe discovers a lot of interesting things coming from different datasources (via the `owl:sameAs` property): directors such as Woody Allen that have been influenced by Kubrick as well as movies such as Zelig. Apart from being graphically visible in the GUI, these pieces of information are available on the form of RDF triples, which can be easily included in his Web page. Once in a while, Joe relaunches the script and gets fresh information directly from the data sources. Besides, he passes the script to his friend Syd that with a slight modification (he changes the seed URI) "centers" the information finding around Alejandro Jodorowsky.


Movies directed by at least one director, more that 50 years old, influenced by Stanley Kubrick.

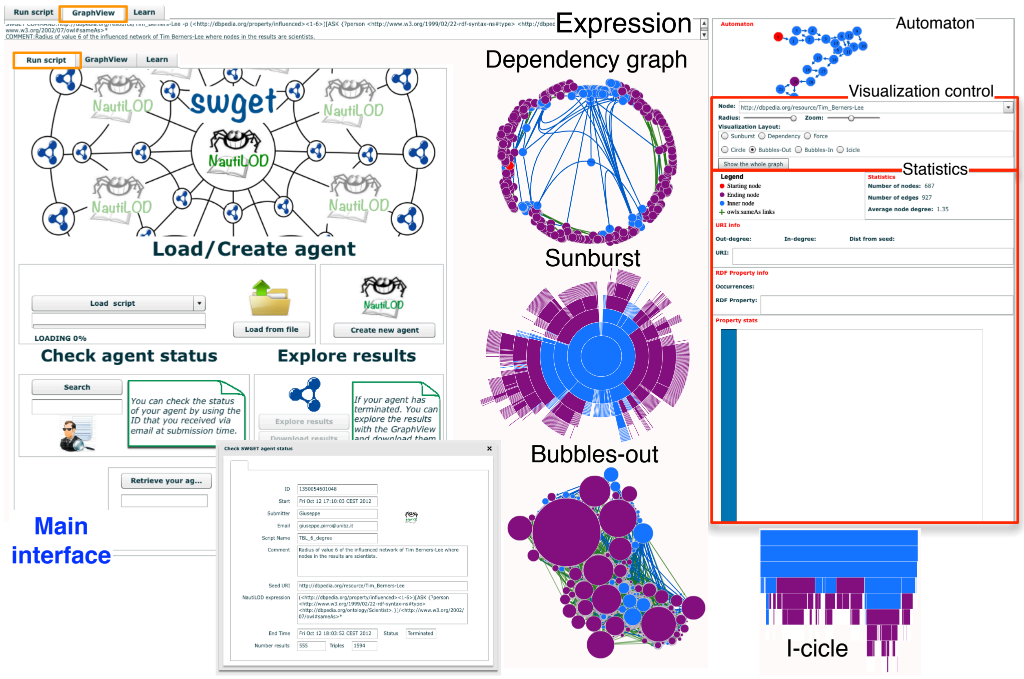
seed URI: http://dbpedia.org/resource/Stanley_Kubrick

NautiLOD expression:

```
<dbpo:influenced>[Q1]/<dbpo:director>/<owl:sameAs>*
Q1=ASK {?person <dbpo:birthDate> ?y.
FILTER(?y < "1962-01-01").}
```

Fig. 5. The SWGET GUI.

Scenario 2 (Web interface): Valerie is a scientific journalist and is writing an article about the Semantic Web and in particular about the figure of Tim Berners Lee (TBL) and his cooperation with other researchers. She thinks that it would be nice to investigate the influence of TBL over other scientists and also from whom he has been influenced. Moreover, having a *reconstruction* of this network and not only a set of "disconnected" nodes would be even more useful. Since Valerie is particularly interested in the scientific community, she thinks that it would be nice to restrict the network to scientists only. Then, Valerie thinks about where to find this information; there are different data sources that may help such as DBPedia and Freebase. Now the problem is how to gather in a clever and automatic way relevant information from different data sources and present it in an attractive way. Fortunately, Valerie is aware of SWGET and by visiting the Web site she discovers that it is available both as a standalone application and a Web based interface available at `swget.inf.unibz.it`. That's what she was looking for! When she runs her SWGET script she is given an *agent id* and gets notified via email as soon as the application has finished. Results according to different visualizations are shown in the right part of Fig. 6. The left parts shows the main interface.



```
SWGET COMMAND:http://dbpedia.org/resource/Tim_Berners-Lee -p (<http://dbpedia.org/property/influenced><1-6>)[ASK {?person <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpec
www.w3.org/2002/07/owl#sameAs>*
COMMENT:Radius of value 6 of the influenced network of Tim Berners-Lee where nodes in the results are scientists.
```

You can check the status of your agent by using the ID that you received via email at submission time.

If your agent has terminated. You can explore the results with the GraphView and download them

Check SWGET agent status

ID	1350054601048
Start	Fri Oct 12 17:10:03 CEST 2012
Submitter	Giuseppe
Email	giuseppe.pirro@unibz.it
Script Name	TBL_6_degree
Comment	Radius of value 6 of the influenced network of Tim Berners-Lee where nodes in the results are scientists.
Seed URI	http://dbpedia.org/resource/Tim_Berners-Lee
NautiLOD expression	(<http://dbpedia.org/property/influenced><1-6>)[ASK {?person <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Scientist>.}]/<http://www.w3.org/2002/07/owl#sameAs>*
End Time	Fri Oct 12 18:03:52 CEST 2012
Status	Terminated
Number results	555
Triples	1594

Automaton

Node: http://dbpedia.org/resource/Tim_Berners-Lee
Radius: Zoom:
Visualization Layout:
( ) Sunburst ( ) Dependency ( ) Force
( ) Circle [✓ Bubbles-Out] ( ) Bubbles-In ( ) Icicle
Show the whole graph

Legend
● Starting node
● Ending node
● Inner node
+ owls:sameAs links

Statistics
Number of nodes: 687
Number of edges 927
Average node degree: 1.35

URI info
Out-degree: In-degree: Dist from seed:
URI:

RDF Property info
Occurrences:
RDF Property:

Property stats

Fig. 6. The SWGET online portal.

Learnt lesson. The aim of SWGET is to give the power to Web users to directly access information. i) However, dereferencing many URIs can be time consuming. We tried to address this issue by the SWGET online portal (`swget.inf.unibz.it`), where users can compose SWGET script and be notified about the results via email. The next step will be to implement these ideas in an agent based infrastructure where the NautiLOD language will be used to instruct mobile agents to navigate the datasources and perform actions. ii) Datasources are noisy. This is especially true for those automatically converted in RDF. iii) It is very difficult to reach small datasources if these are not connected to some "hubs".

Table 1. Appendix: Compliance with Minimal (M) and Additional (A) Requirements

M1) The application has to be an end-user application
swget is available both as a standalone tool and a Web portal ready to use.
M2.1) Information sources under diverse ownership or control
swget scripts span different and autonomous RDF data sources available on the Web of Data.
M2.2) Information sources heterogeneity
The tool is not bound to any particular type of source. Regarding syntax of the sources, it works with RDF and HTML.
M2.3) Information sources and substantial quantities of real world data
All data handled by swget are taken directly from the data source. The information horizon of swget is the Web.
M3.1) Meaning must be represented using Semantic Web technologies.
swget is about *semantic* navigation. It leverages RDF predicates to enables the navigation. Besides, the usage of ASK SPARQL query provide a means to orient the navigation by filtering out certain datasources. The output of scripts are RDF documents that can be further exploited.
M3.2) Data manipulation/processing in interesting ways
swget takes a semantic specification in the NautiLOD language and scrutinizes distributed RDF datasources to discover portions of the Web conform to the specification.
M3.3) This semantic information processing has to play a central role
swget is all about semantics: it uses both RDF and SPARQL to drive and control the navigation.
A1) Web interface
We provide a Web interface implemented in Adobe Flash at `http://swget.inf.unibz.it`.
A2) Scalability
The "horizon" in which swget operates is the whole Web. swget provides a mechanism based on ASK SPARQL queries to control the navigation. For scripts that may take long time, the Web interface enables to submit *jobs* and be notified then the task is completed.
A3) Rigorous evaluations
We performed an evaluation of the NautiLOD language in our WWW2012 paper.
A4) Novelty
swget is a tool to write declarative navigational expression that exploit the semantics of RDF datasources at a Web scale. Besides, it incorporates a mechanism to command actions over data.
A5) Beyond pure information retrieval
swget handles information at a semantic level available in RDF triples.
A6) Commercial Potential and/or large existing user base.
Applications of swget can include: intelligent semantic crawling to build personalized search engine, incorporation of swget scripts in HTML page to enhance their content and so forth.
A7) Contextual information is used for ratings or rankings.
swget accesses data "as it is". It returns exact information from structured sources, thus the notion of ranking and approximation do not apply here.
A8) Multimedia Data
swget can focus on multimedia documents (e.g., images) by exploiting specific predicates (e.g., `foaf:image`) in scripts.
A9) Dynamic Data
swget scripts are executed on the live Web, therefore always guarantee fresh data.
A10) Result accuracy
The user can specify how deep (and wide) would like to search the Web. Over static and reliable data sources swget gives exact results. Over dynamic (and unreliable) data sources on the Web has all features/problems of getting fresh data.
A11) Multiple languages
As data providers (e.g., DBPedia) start to provide multilingual information, it naturally adapts; expressions can be used to filter out information in a particular language.