Artificial Intelligence 239 (2016) 143–167

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Building knowledge maps of Web graphs

Valeria Fionda [a], Claudio Gutierrez [b], Giuseppe Pirrò [c,*]

[a] DeMaCS, University of Calabria, Italy
[b] DCC, University of Chile, Chile
[c] ICAR-CNR, Rende, CS, Italy

ARTICLE INFO

Article history:
Received 22 May 2014
Received in revised form 3 July 2016
Accepted 11 July 2016
Available online 18 July 2016

Keywords:
Maps of the web
Navigation
Web of data
Linked data
Semantic web

ABSTRACT

We research the problem of building knowledge maps of graph-like information. There exist well-consolidated cartographic principles and techniques for mapping physical landscapes. However, we live in the digital era and similarly to the Earth, the Web is simply too large and its interrelations too complex for anyone to grasp much of it through direct observation. Thus, the problem of applying cartographic principles also to digital landscapes is intriguing. We introduce a mathematical formalism that captures the general notion of map of a graph and enables its development and manipulation in a semi-automated way. We present an implementation of our formalism on the Web of Linked Data graph and discuss algorithms that efficiently generate and combine (via an algebra) regions and maps. We present the MaGe tool, implementing the map framework, and discuss examples of knowledge maps.

1. Introduction

The Web can be seen as a vast space of interconnected information that users commonly access via navigation enabled by browsers. However, the Web is simply too large and its interrelations too complex for anyone to grasp much only by direct observation. Consider the task of navigating a citation network by using, for instance, Google Scholar.[1] One typically starts from a seed paper. By clicking on the *cited by* link, one navigates towards papers that have cited the seed paper, selects those of interest (e.g., by bookmarking them) and then continues. After a while it is very hard to reconstruct the network of citations in terms of papers of interest and connections between them. Moreover, the whole process is manual. Having an automatic way of identifying the portion of the citation network of interest (i.e., papers and their connections) and then some form of abstract representation, where only salient papers (e.g., papers with certain keywords in the title) and links between them are represented, is an extremely useful support to the navigation.

To cope with the huge amount of interconnected information available on the Web, we take inspiration from cartography and introduce a framework to build *maps of the Web*. In the physical space, the process of map making can be summarized in two main steps, that is, *selection* and *abstraction* [1]. Selection enables one to focus only on the particular pieces of information that will serve the map's purpose; specifically, in this phase *the region* to be mapped is chosen. In our previous example about navigating a citation network, the region would consist in nodes (i.e., papers) and *cited by* links visited during the navigation. Abstraction is the fundamental property of a map, which states that a map is always smaller than the region

* Corresponding author.
E-mail addresses: fionda@mat.unical.it (V. Fionda), cgutierr@dcc.uchile.cl (C. Gutierrez), pirro@icar.cnr.it (G. Pirrò).

[1] http://scholar.google.com.

http://dx.doi.org/10.1016/j.artint.2016.07.003
0004-3702/

it portrays. Abstracting the region visited while navigating our citation network could be done by considering only nodes with certain properties (e.g., papers published in some specific conference) and links between them.

We see Web Maps as useful navigational cues and powerful ways of conveying complex information via abstract representations; in our context abstraction is used to remove unwanted details that are out of the scope of the map's subject. This gives Web Maps the role of *navigational charts* that help cope with the size of the Web (cyber) space and elude the "lost in the cyberspace" syndrome [2]. Thanks to Web Maps, users can explore complex digital territories, find routes toward destinations and discover previously unknown connections between knowledge items. Web Maps are also useful for analyzing information. For instance, the availability of a series of chronologically sequential maps enables complex map analysis (e.g., longitudinal analysis) for the detection and forecasting of trends in specific domains. This is useful, for instance, in the analysis of knowledge flows in scientific literature to show how the interlinking between disciplines is changing [3]. Another example are maps of social networks that can be analyzed to forecast friendship [4].

Recent progress in Web technologies and languages originating from the Semantic Web proposal *as well as* the availability at planetary scale of structured information in the Resource Description Framework (RDF) standard data format, open new opportunities *toward automating* the construction of Web Maps. On one hand, interlinks between data items, encoded in RDF predicates, carry a precise semantic meaning, thus allowing for precise characterization of the nature of reachability that *is crucial* in extracting regions of the Web. On the other hand, maps can be given an RDF representation and then be processed not only by humans, via visual interpretations, but also by machines, due to the machine-processable nature of RDF. This will foster the exchange, combination and reuse of maps. We believe that the availability of Web Maps can help human users cope with the complexity of Web Regions in the same way as geographic maps help users cope with the complexity of large physical regions.

Notions of maps for digital landscapes have been around for some time. Doemel [5] introduced the idea of Web Map as a means to capture user navigational activities. This kind of map is not a map in the cartographic sense as it misses the abstraction phase, which is the raison d'être of a map [1]. There are also many tools (partially) touching upon the problem of building maps of the Web. The most traditional and popular are bookmarks: a list of URLs specified by a user when navigating the Web. This idea has been enhanced to incorporate, for instance, social features (share, rank, tag) and/or annotations of different types of data (e.g., not only pages but also documents). Delicious,[2] Diigo,[3] and Google Bookmarks[4] are among the most popular bookmarking systems. Some systems go beyond simple bookmarks by enabling one to organize URLs to also highlight connections between *the two*. Results are grouped and presented in the form of a graph, which simulates the idea of a region of the Web. Examples of such systems are search engines like Tag Galaxy,[5] navigational history tools (e.g., [5,6]), visual HTML site maps (for users) and atlases of the Web (e.g., [2]). More recent approaches focus on providing visual representations of information in specific domains such as publications or news (e.g., [7,8]).

Existing approaches, discussed in Sections 2.1 and 6, do not comply with the idea of a map that we envision. First, even if they partially simulate the idea of capturing regions of the Web, they do not consider the abstraction of Web Regions to build maps. Second, they are designed for human visualization; hence their automatic processing, composition and reuse are not considered, which hinders the exchange, automatic combination and interpretation of maps. Third, they do not enable the declarative specification of the region (e.g., portion of interest of the Web) to be mapped thus hindering the automation of the process of creating maps. Fourth, they lack a formal mathematical model. They do not guarantee formal/provable (reachability) relations between the points in the map; formal notions of granularity and scope; and formal provable relations between the map and the region it represents. These limitations obstruct the generation of formal deductions from maps.

1.1. Contributions

In this paper we formally introduce the notion of Web Map and face several challenges toward this goal. First, given a user need or a conceptual notion, *provide a way to specify* a region of the Web that represents or encompasses it. Second, given a region of the Web, define what is a map of it and investigate its formal properties. Third, devise algorithms and compose the procedures efficiently. The contributions of this paper are:

- We provide a mathematical formalization of the notions of region and map of a graph; we discuss several types of maps, present algorithms for constructing such maps and study their complexity.
- We introduce an algebra for maps and study its formal properties.
- We discuss the problem of obtaining regions of the Web via graph navigational languages; to this aim we introduce a general navigational language to specify Web Regions.
- We showcase an implementation of the formal map framework and navigational language on the current Web of Linked Data via a tool called MaGe (Map Generator).[6]
- We discuss some examples of Web Maps with real data.

[2] http://delicious.com/.
[3] http://www.diigo.com/.
[4] http://www.google.com/bookmarks/.
[5] http://taggalaxy.de/.
[6] The tool is available at the Website http://mapsforweb.wordpress.com.

An overview of the idea of Web Maps was given in Fionda et al. [9,10]. The present paper significantly extends our previous work by providing: *(i)* all the formal definitions *(ii)* a new family of maps (i.e., k-maps); *(iii)* all the formal proofs; *(iv)* algorithms and complexity; and *(v)* further examples of maps.

1.2. Organization of the paper

The paper starts with a general overview of the problem of building Web maps and continues with a formal framework to construct maps of a graph (Section 3). We present efficient algorithms to compute maps (Section 3.2) and an algebra for maps (Section 3.4). Section 4 discusses how navigational languages can be used to automatically extract regions of interest from a graph. We discuss the implementation of the MaGe tool to build Web Maps in Section 5. Here, we also provide concrete examples to show the feasibility and usefulness of Web Maps. We perform a detailed review of related work in Section 6. We present conclusions in Section 7.

2. Problem statement: why maps of the Web?

The Web is a huge information space of interconnected items commonly accessed via navigation. To cope with the size and complexity of the Web we introduce the idea of Web Maps that help users track, record and identify conceptual regions of information on the Web for their own use, for sharing/exchanging with other users and/or for further processing (e.g., in combination with other maps). We now provide some informal definitions underlying the idea of maps that will be formalized in Section 3.

Definition 2.1 *(Region[7]).* An area of a given space with some characteristics that distinguish it from other areas. A region usually represents a territory of interest to people and for which one or more distinctive traits are used as the basis for its identity.

Definition 2.2 *(Map[7]).* A map is a (visual) abstract representation of a region; a symbolic representation highlighting relationships between some elements in the region.

We now introduce a concrete scenario that illustrates the idea of Web Maps.

Example 2.3 *(Mapping favorite directors).* Syd is fond of cinema and wants to find his favorite directors. He imagines that a convenient way to keep track of such directors and their relations, would be to put them in a map. The map will represent connections among Syd's favorite directors, in a concise way. Syd is a friend of Joe that is also fond of cinema and already has a map of his favorite directors. Syd wonders whether he can import the map of Joe's favorite directors to his computer and whether his and Joe's map can be combined (e.g., intersected).

Several research challenges emerge in relation to the development of Web Maps as those discussed in Example 2.3. First, the Web is huge, widely distributed and continuously changing, therefore techniques to specify, access and retrieve parts of it, that is, regions of interest are needed. Second, given a region of the Web, what is a reasonable formal definition of a map? How can we efficiently build Web Maps? What are the challenges in automating the construction and combination of Web Maps? Besides these aspects, that project the principles of *selection* and *abstraction* of traditional cartography to the Web, additional research challenges emerge such as the possibility of going beyond maps for human users only. This sets the need for a mathematical model of regions and maps of the Web so that their properties and mutual relationships can be rigorously defined and understood. As anticipated earlier, there is no comprehensive and principled solution to the problem of obtaining maps as those described in Example 2.3. The main limitations of current solutions are summarized as follows (see Section 6 for a detailed discussion):

- *No automatism*: current approaches do not enable automation. The user has to *navigate* the Web to identify relevant information; it is not possible to declaratively express a conceptual need and automatically retrieve relevant information.
- *No connections*: most systems do not maintain connections among pieces of information of interest. However, reachability information is crucial for understanding complex relations among data items and also for building maps.
- *No abstraction*: current approaches do not consider abstraction, which is the fundamental property of a map. Abstraction enables the representation of a region of interest from the Web, in a concise way.
- *No automatic processing*: it is not possible today to automatically generate, process and combine maps of the Web.
- *No formal framework*: there is no formal framework that addresses the problem of extracting regions of the Web, builds maps and combines them via an algebra.

[7] Definition adapted from Wikipedia.

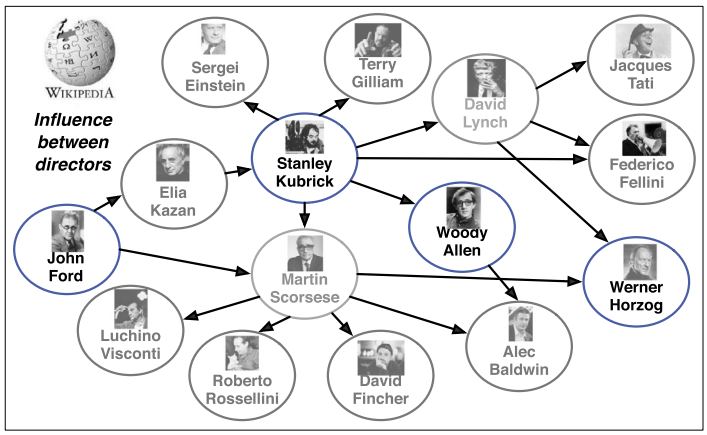

Fig. 1. Region tracked by Syd while navigating and marking Wikipedia pages about his favorite directors. The starting point is the Wiki page about J. Ford.

In the remainder of this section, by using Example 2.3 as a reference, we give an overview of the idea of map of a Web Region and introduce the main research challenges that will be developed in the subsequent parts of the paper. In particular, the objective of the example is to show *what* can be done *manually* today in the Web of documents (i.e., Wiki pages) in terms of selecting interesting portions of it (e.g., favorite directors) and how this process can be *automated* via the framework we propose.

Toward web maps. The construction of a map in a given information space comprises two main steps, that is, *selection* and *abstraction*. Our focus is the Web space, a huge relational space where reachability is a fundamental property. Reachability enables *navigation*, the common operation that both human (Web users) and machines (via crawling) perform on a daily basis to reach information of interest in some region of the Web space. Indeed, given a conceptual specification (e.g., Syd's favorite directors) Syd can *manually* traverse links and mark nodes (Wiki pages) of interest to track a Web Region. Links are unlabeled (they do not carry semantics) as the semantics is encoded in the text that the user reads to decide whether the link has to be followed (if Syd decides that the link represents an influence relation between directors). Fig. 1 shows a hypothetical region of the Web obtained from Wikipedia as the result of the navigational activity of Syd (see Example 2.3) who, starting from one of his favorite directors (i.e., J. Ford) and traversing *influence* links, has manually (book)marked other favorite directors, that is, S. Kubrick, W. Allen and W. Herzog. Note that the region besides these *distinguished* nodes also contains other (lighter) nodes that were visited during the navigation. For instance, the node E. Kazan was visited when navigating from J. Ford to S. Kubrick.

Effectively, this region resulted from a conceptual specification that was in the mind of Syd, that is, *finding his favorite directors (in)directly influenced by J. Ford*. The first research question that arises regarding the development of the notion of map of a Web Region is the following: is it possible to provide a language to declaratively express such kinds of specifications? Such a language along with an automatic mechanism for accessing information on the Web will alleviate Syd's burden; there will be no need for him to manually navigate and track the region of interest in the Web and select relevant information. Syd could use the language to specify his favorite directors, singers or more complex things like his favorite soccer players that currently live in Italy and are younger than 30 years old. Answering the research question above corresponds to automate the *selection* phase in traditional cartography. We will show in Section 4 how to achieve this goal via graph navigational languages.

To cope with the phase of *abstraction* in traditional cartography, which is meant to hide unwanted details from a region, the second research question arises: given the region of the Web in Fig. 1, what does a map of Syd's favorite directors look like? To answer this question, there is the need to give maps a mathematical characterization, which will also enable representation, exchange, reuse and combining of maps with well-defined operations like union and intersection. Such a formalism will enable Syd to combine (see Example 2.3) his map with Joe's map, thus making sense of their commonalities and differences. We discuss this aspect in the general map framework in Section 3.

Fig. 2 shows two possible maps of the region in Fig. 1. Map 1 contains more nodes (e.g., M. Scorsese) and edges (e.g., the edge e_1) than Map 2. Map 2 adopts a specific abstraction strategy; it preserves reachability information *only* among pairs of distinguished nodes. The node M. Scorsese in Map 1 is not present in Map 2 since it is not a distinguished node. Note that in Map 2 the reachability between J. Ford and W. Herzog (both distinguished nodes) is encoded via the edge e_2. The edge e_1 in Map 1 is not present in Map 2; note that the only path between J. Ford and W. Allen in the region is J. Ford→E. Kazan→S. Kubrick→W. Allen, which is encoded in e_1. This path passes through S. Kubrick, which is a distinguished node and then must be in the map along with the edges J. Ford→S. Kubrick (representing the path J. Ford→E. Kazan→S. Kubrick) and S. Kubrick→W. Allen. Reachability information therefore encoded in e_1 in Map 1 is already represented in the path J. Ford → S. Kubrick → W. Allen in Map 2; hence e_1 can be discarded. We formalize this reasoning in the framework to build maps of graphs in the following sections.

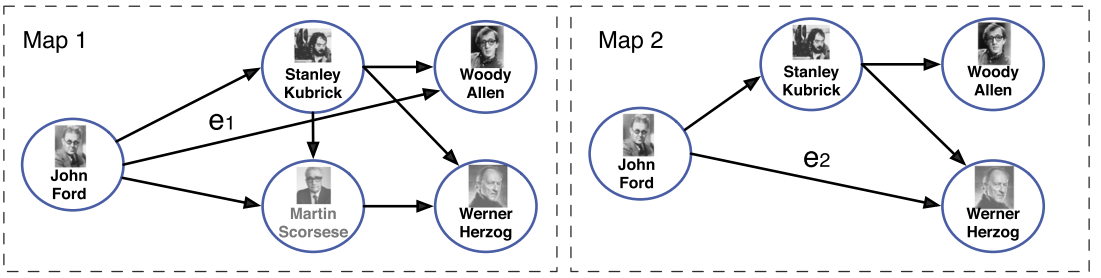

Fig. 2. Two possible maps of the region in Fig. 1.

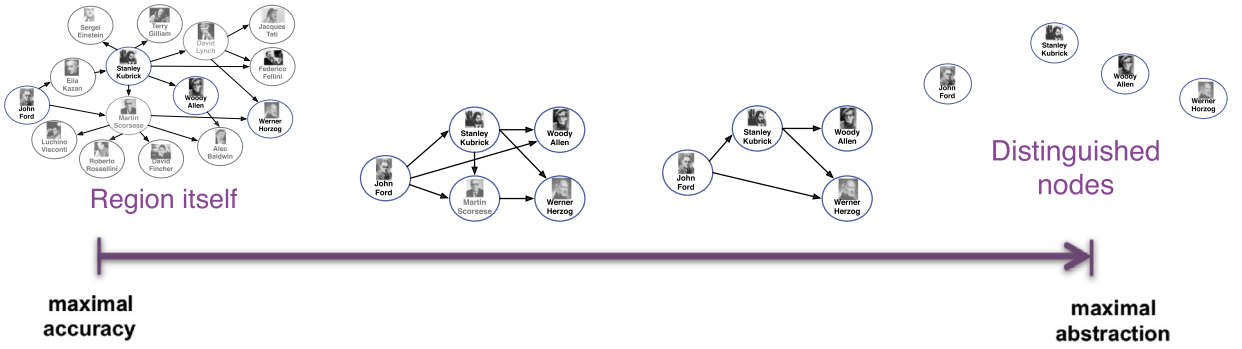

Fig. 3. Different levels of abstraction.

Note that in traditional cartography the process of abstraction is obtained via the map *scale*, which sets the ratio of the distance between two points on a map and their real distance. The dimensional relationship between the region and the map is fixed; hence all the points in the region usually are of equal importance. The idea of Web Maps discussed above is different in the sense that it abstracts the region on the basis of some specific points of interest (i.e., Syd's favorite directors) that must be in the map. Therefore, *reachability* and *navigation* are two essential features to consider when building maps of the Web space.

With these considerations in mind, the idea of a map of a Web Region is essentially a *concise way* of reflecting information in the region in terms of *reachability* among particular distinguished nodes. At this point the problem arises of how much of the original region (besides the distinguished nodes) has to be included in the map (see Fig. 3). Robinson et al. [1] state that abstraction gives the map its raison d'être. Indeed, the writer J.L. Borges scoffs at perfectly accurate maps when he talks about a "map of the Empire whose size was that of the Empire, and which coincided point for point with it" [11]. At the other extreme, minimal maps contain only some reference points without any relation between them (e.g., bookmarks on the Web). We believe that a flexible mapping framework should consider different types of maps (see Section 3.1). The above discussion is the starting point for building maps of Web regions. The last research question (tackled in Section 5.1) is how to instantiate the map framework on the current Web infrastructure in order to automate the construction of maps.

2.1. Feasibility of web maps

With the advent of semantic data at a massive scale originating from the Semantic Web proposal, maps to describe and navigate information on the Web in a machine-processable way become feasible. The key new technical support includes the increasing availability of structured data and tools to access it. Projects such as Linked Open Data [12] aim to publish and interlink information by using a standard semantic infrastructure with the Resource Description Framework (RDF) as the main pillar. To access this data there is a standard query language called SPARQL [13], and various navigational languages (e.g., [14,15]) that leverage the graph-nature of RDF data. The aim of this paper is to enable the creation of maps of Web Regions that are machine-processable, that is, with a well-defined semantics, representable in RDF to facilitate their interchange, endowed with provable formal properties, reusable, that can be composed, and of course, feasible to be constructed. In order to reach this goal we use the following ingredients:

- Since the Web can be conveniently modeled as a graph, we leverage formal techniques from graph theory to generically define the notion of region and map of a graph with reachability as the main pillar. Our formalism also provides an algebra for maps.
- We leverage navigational languages as a means to declaratively specify and retrieve regions of interest from a graph.
- We instantiate and implement the map framework, that is, the definition of map, the algebra of maps and the navigational language on the Web of Linked Open Data [16].

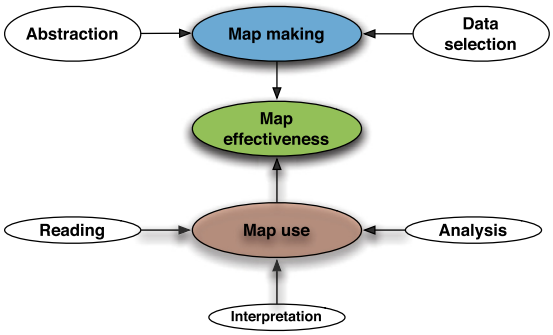

Fig. 4. Designing of Web maps. Adapted from [1].

Summarizing all the machinery described so far, the high level specification for building maps of the Web is:

1. *Specify the resources of interest*: this can be done in different ways. In most cases the specification is not formal; it is a collection of nodes of the Web graph (i.e., URLs), given by user experience, social recommendations, output of search engines, etc. This has a drawback: there is no high level formal description of such a set of nodes. In this paper we use a general graph navigational language that incorporates different kinds of tests. One could alternatively use SPARQL [13] or any other language to describe and query RDF data. We discuss these aspects in Section 4.
2. *Build the region corresponding to this specification:* as we will review in Section 6, there are many forms of constructing graphs from a set of nodes. However, most of these procedures are based on statistical methods. Thus, combining, extending or updating the objects they deal with is difficult today. In this paper we use the Web Region visited when evaluating an expression in the navigational language and discuss these aspects in Section 5.
3. *Build a formal map corresponding to the region:* currently there are many proposals to construct maps of graphs, but none of them is able to formally specify the map and establish precise relations between the map and the region. This impedes the combination, reuse and derivation of properties of maps. We discuss these aspects in Section 3.

3. Maps as mathematical objects

To tackle the complexity of the huge amount of information interlinked on the Web, we take inspiration from cartography, which provides a set of principles (reported in Fig. 4) that are useful for introducing a rigorous notion of Web Map. Cartography [1] relies on the mind's ability to read complex information represented in the map; hence, visualization plays a fundamental role. Our challenge is to go beyond human-readable maps and enable map interpretation also by machines. Toward this goal there is the need to give maps a precise mathematical definition and investigate their properties and relations with the regions from which they come. Finally, maps can be concretely represented in the RDF format, which is suitable for their exchange and unambiguous interpretation.

Map Making includes two main steps, that is, *selection* and *abstraction*. The first is used to select data that will be put inside a map; selecting the region of the space to be mapped. This aspect will be extensively discussed in Section 4. The second one deals with the reduction of the amount of information in the region that will finally form the map. Map usage includes different options such as analyzing and interpreting information in the map. We want to point out that although we consider cartographic principles for geographic maps, our research goal has the following peculiarities: *(i)* maps have to be suitable for usage not only by humans but also by machines; *(ii)* traditional cartography builds upon the notion of Euclidean space, while in our context, the space being mapped, that is the Web, is purely relational; *(iii)* traditional cartography uses the notion of scale to abstract information on a map; the scale gives all points equal importance. Our idea of abstracting information in a Web Region is different: we start with a set of points of interest that must be in the map (e.g., Syd's favorite directors) and build the map around them. Our notion of abstraction is related to the nodes that are included in the map apart from those marked a distinguished.

3.1. A formal definition of map

In order to introduce our map framework we model the Web space as a directed graph. As will be discussed in Section 4, we use a navigational language to automatically extract regions of the Web. The benefit of the framework we are going to present is that maps have formally defined properties and can be effectively treated as mathematical objects. We now introduce some notation and some basic definitions. Let $\Gamma = (V_\Gamma, E_\Gamma)$ be a Web Region, where V_Γ and E_Γ are the set of nodes and directed edges, respectively. In the process of map constructions, we ignore edge labels (if present) as we are interested in capturing reachability between nodes. Fig. 5 shows an example of Web Region. In the remainder of the paper, we use the following notation:

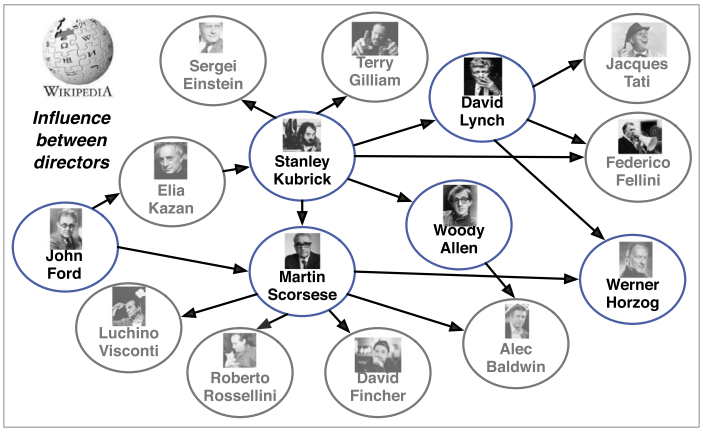

Fig. 5. A Web Region with some distinguished (blue) nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

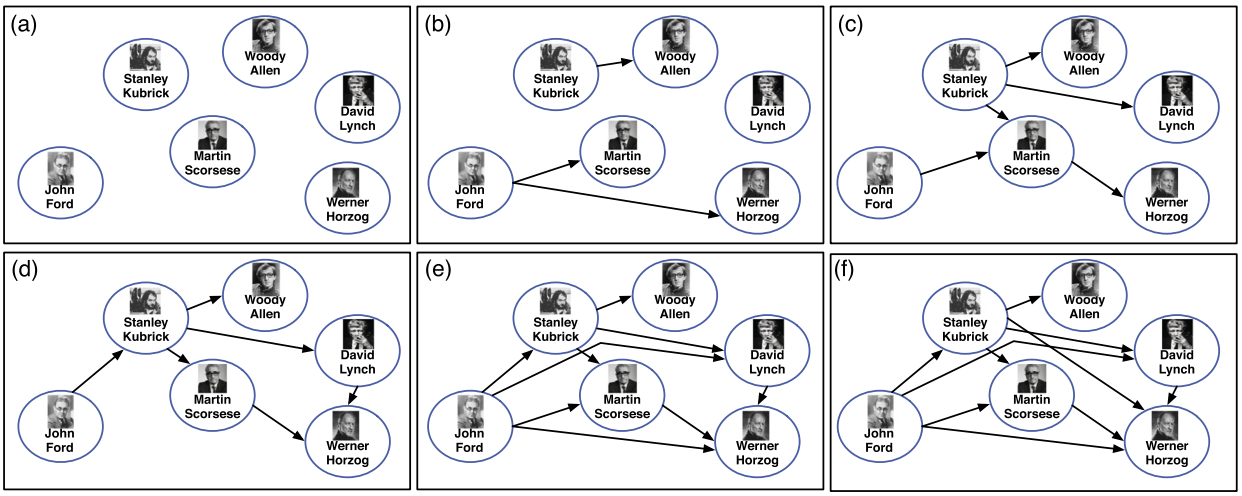

Fig. 6. Examples of maps.

- $u \to v$ denotes an edge $(u, v) \in E_\Gamma$. For instance, in Fig. 5 we have $S.\ Kubrick \to W.\ Allen$.
- $u \twoheadrightarrow v$ denotes a path from u to v in Γ. In Fig. 5 we have $J.\ Ford \twoheadrightarrow S.\ Kubrick$.
- Let $N \subseteq V_\Gamma$, we write $u \twoheadrightarrow_N v$ if and only if there is a path from u to v in Γ not passing through intermediate nodes in N. As an example, if $N = \{M.\ Scorsese, W.\ Allen\}$ by considering the region in Fig. 5 we have that $J.\ Ford \twoheadrightarrow_N S.\ Kubrick$ but not $J.\ Ford \twoheadrightarrow_N A.\ Baldwin$.

Definition 3.1 *(Map).* A *map* $M = (V_M, E_M)$ of $\Gamma = (V_\Gamma, E_\Gamma)$ is a directed graph such that $V_M \subseteq V_\Gamma$ and each edge $(x, y) \in E_M$ implies $x \twoheadrightarrow y$ in Γ.

The idea of a map is essentially that of representing reachability information between pairs of *distinguished* nodes, in a concise way, that is, nodes in the set $V_M \subseteq V_\Gamma$. By making a parallel with geography, Γ represents the "region" or "territory" being abstracted (see also Definition 2.1) via the map and nodes in V_M represent "points" that are relevant for the map; these are the *distinctive traits* used as the basis for the *identity of the region* (see Definition 2.2). For example, distinguished nodes can be our favorite directors in a graph of directors or relevant scientific papers in a graph of bibliographic data. Distinguished nodes alone capture a very basic and common form of map of the Web, that is, bookmarks. In this case, the region is the portion of the Web graph navigated to reach the distinguished nodes, that is, the pages bookmarked.

Fig. 6 shows some possible maps obtained from the region in Fig. 5 where blue nodes are the distinguished nodes. The map in Fig. 6(a) represents an example of bookmarks, that is, nodes that have been marked as interesting.

Note that maps (a)–(c) do not capture reachability among distinguished nodes. For instance, J. Ford and S. Kubrick are both distinguished nodes and they are connected by a path in the region, however, they are not connected in these maps. This leads to the definition of a complete map.

Definition 3.2 *(Complete map).* A *map* $M = (V_M, E_M)$ of $\Gamma = (V_\Gamma, E_\Gamma)$ is complete if, and only if, $\forall\, x, y \in V_M$, $x \twoheadrightarrow y$ in Γ implies $x \twoheadrightarrow y$ in M.

Complete maps solve the problem of reachability among distinguished nodes. In Fig. 6, maps (d)–(f) are examples of complete maps. However, even completeness is not enough to abstract information. Consider the map in Fig. 6(d). In the region in Fig. 5 there is a direct edge connecting J. Ford and M. Scorsese. Although there is a path connecting the two nodes (via S. Kubrick) in the map, the fact that there is a direct connection between them is missing. Hence, we introduce the notion of route-complete map.

Definition 3.3 *(Route-complete map).* A *map* $M = (V_M, E_M)$ of $\Gamma = (V_\Gamma, E_\Gamma)$ is route-complete if, and only if, $\forall\, x, y \in V_M$ $x \twoheadrightarrow_{V_M} y$ in Γ implies $x \to y$ in M;

The following Lemma shows that route-completeness is a more general notion than completeness.

Lemma 3.4. *Every route-complete map M of Γ is also a complete map.*

Proof. Let $x \twoheadrightarrow y$ in Γ, M be a route-complete map and $x, y \in V_M$. We show that $x \twoheadrightarrow y$ in M holds by induction on the number of nodes in V_M that appear in the path $x \twoheadrightarrow y$ in Γ (not considering x and y).

Base case: if $x \twoheadrightarrow_{V_M} y$ in Γ, then by route-completeness $x \to y$ in M.
Inductive step: assume that $\forall x, y \in V_M$ such that $x \twoheadrightarrow y$ in Γ passes through at most n nodes in V_M then $x \twoheadrightarrow y$ in M. Let $x, y \in V_M$ such that $x \twoheadrightarrow y$ in Γ passes via $n+1$ nodes in V_M and let $z \in V_M$ be a node such that $x \twoheadrightarrow z$ and $z \twoheadrightarrow y$. We have that $x \twoheadrightarrow z$ and $z \twoheadrightarrow y$ both pass through at most n nodes in V_M and thus by induction we have that $x \twoheadrightarrow z$ and $z \twoheadrightarrow y$ in M, thus $x \twoheadrightarrow y$ in M. □

By looking at Fig. 6, map (d) is not route-complete while maps (e) and (f) are route-complete. Indeed, for each path between distinguished nodes not passing through any distinguished node in the region, there is an edge in the map. Nevertheless, complete and route-complete maps also suffer from other problems. If we consider maps (e) and (f), both include some direct edges like, for instance, J. Ford→S. Kubrick although not originally present in the region; this edge *abstracts* the path J. Ford→E. Kazan→S. Kubrick. Consider now the edge J. Ford→W. Herzog in maps (e) and (f). Note that in the region, all paths connecting J. Ford to W. Herzog pass through some distinguished nodes (i.e., M. Scorsese, S. Kubrick, D. Lynch). The edge J. Ford→W. Herzog does not abstract any reachability information corresponding to paths not passing through any distinguished node. Indeed, route-completeness implies that every path in the region not passing through any distinguished node must be abstracted by a direct edge in the map. However, the definition does not prevent the map having direct edges that do not abstract any paths with such feature. This is the case of the edge J. Ford→W. Herzog, for which reachability is already abstracted via the paths J. Ford→M. Scorsese→W. Herzog, J. Ford→S. Kubrick→M. Scorsese→W. Herzog and J. Ford→S. Kubrick→D. Lynch→W. Herzog. To capture such types of behavior we introduce non-redundant maps.

Definition 3.5 *(Non-redundant map).* A *map* $M = (V_M, E_M)$ of $\Gamma = (V_\Gamma, E_\Gamma)$ is non-redundant if, and only if, $\forall\, x, y \in V_M$ $x \to y$ in M implies $x \twoheadrightarrow_{V_M} y$ in Γ.

An example of non-redundant map is reported in Fig. 6 (d). However, recall that (d) does not satisfy the property of route-completeness. In order to capture both route-completeness and non-redundancy we introduce *good maps*.

Definition 3.6 *(Good map).* A *map* M of Γ is good if, and only if, it is route-complete and non-redundant.

The idea behind the notion of good map is that of giving an economic representation of the reachability between pairs of distinguished nodes. Moreover, good maps have an important property as reported in the following theorem.

Theorem 3.7. *Let $\Gamma = (V_\Gamma, E_\Gamma)$ be a Web Region. Given $N \subseteq V_\Gamma$, there is a unique good map M over Γ with V_M=N.*

Proof. Given the set of distinguished nodes N, the existence of an edge $x \to y$ in M is uniquely determined by the existence of a path in Γ connecting x to y. □

Fig. 7 shows the unique good map of the region in Fig. 5. A flexible map framework should consider maps ranging from perfectly accurate (regions themselves) to minimal. Good maps are minimal maps where reachability information is also included. However, in some cases one would need to include more nodes besides the distinguished nodes included in good maps. To provide a flexible map framework, we introduce *k-maps*, a family of good maps that select nodes in the region by considering specific properties of the nodes. To this end, let $f : V_\Gamma \to \mathcal{R}$ be a real-valued function defined over the nodes

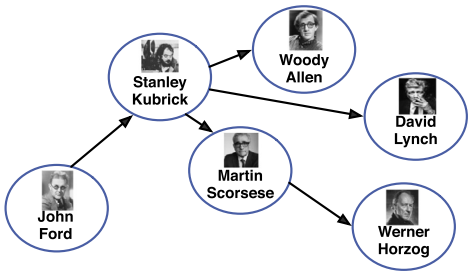

Fig. 7. The unique good map of the region in Fig. 5.

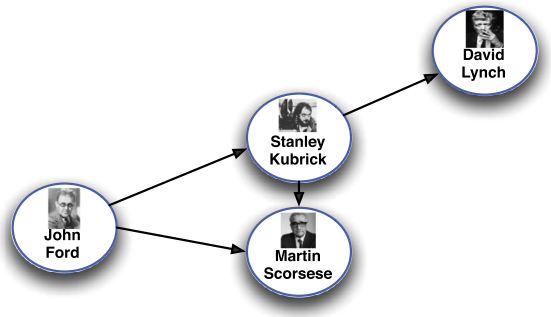

Fig. 8. A 2-map of the region in Fig. 5.

of the region. We say that f is *selective* if the property $\{v \mid v \in V_\Gamma \text{ s.t. } f(v) \geq a\} \subseteq \{v \mid v \in V_\Gamma \text{ s.t. } f(v) \geq b\}$ holds for every pair $a, b \in \mathcal{R}$ s.t. $a \geq b$.

Definition 3.8 *(k-Maps).* Let $\Gamma = (V_\Gamma, E_\Gamma)$ be a Web Region. The k-map of Γ is the good map generated by the set of distinguished nodes $\{v \in V_\Gamma : f(v) \geq k\}$, where $f : V_\Gamma \to \mathcal{R}$ is a selective function measuring a specific property of the nodes in Γ.

The function f can be, for instance, a measure of the centrality of nodes (e.g., PageRank, closeness centrality), a popularity measure (e.g., number of incident edges) and so forth. k-Maps allow for the building of maps at different levels of granularity (in terms of number of nodes) as they allow for the potentially addition of other nodes, apart from the distinguished ones, that satisfy the property defined by the function f. As an example, if we consider f to be the number of outgoing edges, a 2-map of the region in Fig. 5 is shown in Fig. 8. Note that k-maps preserve the property of uniqueness, and are indeed good maps. However, the set of nodes that must be included in the map is not given a priori but is determined via f. The following proposition links good maps and k-maps.

Proposition 3.9. *Let Γ be a Web Region and $n \geq k \geq 0$ two integers. If M_k is the k-map and M_n is the n-map of Γ built using the same function f, then M_n is a good map of M_k over the set of distinguished nodes V_{M_n}.*

Proof. M_n is a map of M_k since $V_{M_n} \subseteq V_{M_k}$. Now we have that $x \to y$ in M_n if, and only if, $x \twoheadrightarrow_{V_{M_n}} y$ in Γ if, and only if, $(x \twoheadrightarrow_{V_{M_k}} y$ in Γ or $x \twoheadrightarrow y$ in Γ passes through some nodes v for which $k \leq f(v) < n)$ if, and only if, $(x \to y$ in M_k or $x \twoheadrightarrow_{V_{M_n}} y$ in $M_k)$. □

3.2. Computing good maps

Maps capture information in a region given a set of nodes. When it comes to their computation, a question arises: what is the estimated size of a map? Fortunately this question has an easy answer: given a set of distinguished nodes V_M, the number of edges of M could be in the worst case $|V_M|^2$. Consider a region Γ being a (directed) clique. If $V_M = V_\Gamma$ then $M = \Gamma$; otherwise, since the map M is route-complete, for every pair of nodes $u, v \in M$, there is a $z \in V_\Gamma \setminus V_M$ such that $u \to z \to v$ in Γ. Note that, depending on the set $V_M \subseteq V_\Gamma$ of node selected as distinguished, it may be the case that the number of edges in the map can be larger than the number of edges in the region, i.e., $|E_M| \geq |E_\Gamma|$. However, according to the above reasoning we have that $|E_M| \leq |V_\Gamma|^2$ holds.

The second question is: how costly is it to compute a map? It depends on what type of graph is the region. Below we show algorithms for computing good maps of Web Regions represented as general directed graphs and the important case of directed acyclic graphs useful for representing particular kinds of Web Regions like citation networks.

Theorem 3.10. *Let $\Gamma = (V_\Gamma, E_\Gamma)$ be a Web Region. Given $V_M \subseteq V_\Gamma$, the unique good map $M = (V_M, E_M)$ of Γ can be computed in time:*

1. $\mathcal{O}(|V_M| \times (|V_\Gamma \setminus V_M| + |E_\Gamma|))$ *if Γ is a general graph by using Algorithm 1.*
2. $\mathcal{O}((|V_M| \times |V_\Gamma \setminus V_M|) + |E_\Gamma|)$ *if Γ is a DAG by using Algorithm 2.*

Proof. We will discuss the two cases separately.

1. The bound can be obtained by adapting the BFS algorithm (see, Algorithm 1). The idea is to compute the BFS starting from each node $v \in V_M$. Each execution of the basic BFS costs $\mathcal{O}(|V_\Gamma| + |E_\Gamma|)$, hence the total cost is $\mathcal{O}(|V_M| \times (|V_\Gamma| + |E_\Gamma|))$. However, since only paths that do not pass through any node in V_M are relevant in building the good map, the edges of Γ can be partitioned in two disjoint sets: $E_\Gamma = E_M \cup E_{\overline{M}}$, where E_M is the set of edges (v, v') s.t. $v \in V_M$ and $E_{\overline{M}}$ is the set of edges (v, v') s.t. $v \notin V_M$. During the BFS, if a node $v \in V_M$ is reached, the corresponding edge is added to the map (line 12) while all the outgoing edges are discarded since v is not added to the queue of the nodes to be visited. Thus, the cost of the BSF becomes $\mathcal{O}(|V_M| \times (|V_\Gamma \setminus V_M| + |E_{\overline{M}}|) + |E_M|)$. This is because edges in $E_{\overline{M}}$ are visited when performing the BFS starting from each node in V_M and edges in E_M are visited only once when the BFS is executed from their originating (distinguished) node. A slight improvement of the running time can be obtained by stopping the BFS from a node as soon as all the nodes in V_M have been visited (lines 4, 7, 14). The space complexity is $\mathcal{O}(|V_\Gamma|)$, the same as BFS.
2. The bound can be obtained by looking at Algorithm 2. Let $|V_M| = N$. The first step (line 1) is to topologically order the nodes in Γ and rename them (line 5) in order, say $v_1, ..., v_n$. The initialization of the data structures (lines 6–7) corresponds to the building of a list of N bits for each node v_i in $V_\Gamma \setminus V_M$ and initializing it with all cells equal to 0. The algorithm then visits nodes in order and proceeds as follows:
   (a) If v_j is in V_M, then for each of the nodes w reached via an edge, we have either: *(i)* if w is not in V_M, mark the bit j in its list as 1; *(ii)* if $w \in V_M$, add the edge (v_j, w) to the map (lines 9–14).
   (b) If v_j is not in V_M, then for each of the nodes w reached via an edge, we have either: *(i)* if w is not in V_M, do the OR of the lists of v_j and w; *(ii)* if $w \in V_M$, then output the edges (v_k, w) for each place k in the list with bit 1 (lines 16–21).

   The edges in E_Γ are used once and the time for managing the list is $\mathcal{O}(|V_\Gamma \setminus V_M| \times N)$. □

Algorithm 1: GoodMap(Region Γ, set of nodes V_M).

Input : A region Γ, a set of distinguished nodes V_M
Output: Good map M

```
1  for all v ∈ V_M do
2    for all u ∈ V_Γ do
3      reach(u) = false
4    num_reached = 1 {only v has been reached}
5    reach(v) = true
6    Q[v] {queue containing only v}
7    while Q is not empty AND num_reached < |V_M| do
8      u = Q.dequeue
9      for all edge (u, w) ∈ E_Γ do
10       if reach(w) = false then
11         reach(w) = true
12         if w ∈ V_M then
13           M.addEdge(v, w)
14           num_reached = num_reached + 1
15         else
16           Q.enqueue(w)
17 return M
```

When dealing with k-maps, the nodes $V_{M_k} \subseteq V_\Gamma$ to be included in the map are obtained by checking the value of f for each node. For example, this can be done in time $\mathcal{O}(|E_\Gamma|)$ when using degree centrality. Algorithm 1 or Algorithm 2 is then executed over Γ considering the set of nodes V_{M_k} as input; from Theorem 3.10, we can derive that k-maps can be constructed in time $\mathcal{O}(|V_M| \times (|V_\Gamma \setminus V_M| + |E_\Gamma|)) + \mathcal{C}_f(V_\Gamma)$ for general graphs and $\mathcal{O}((|V_M| \times |V_\Gamma \setminus V_M|) + |E_\Gamma|) + \mathcal{C}_f(V_\Gamma)$ for DAGs, where $\mathcal{C}_f(V_\Gamma)$ is the cost of computing the value of the function f for all the nodes in Γ.

3.3. Computing maps: a running example

We now describe an example of how to compute good maps by using Algorithm 1. Consider the region Γ in Fig. 9 where the initial state of the data structures used in the algorithm is also shown. The array V_M contains the set of distinguished

Algorithm 2: GoodMapDAG(Region Γ, set of nodes V_M).

Input : A region Γ, a set of distinguished nodes V_M
Output: Good map M

```
1  ordered_nodes=TopologicalOrder(Γ)
2  i = 0;
3  for all v ∈ ordered_nodes do
4     i=i+1
5     associate(v, i)
6     if v ∉ V_M then
7        create_bit_array(v_i, N, 0)
8  for all v ∈ ordered_nodes do
9     if  v ∈ V_M then
10       for all edges (v, w) ∈ E_Γ do
11          if w ∉ V_M then
12             mark(w, v)
13          else
14             M.addEdge(v, w)
15    else
16       for all edge (v, w) ∈ E_Γ do
17          if w ∉ V_M then
18             update_bit_array(w,OR(get_bit_array(v),get_bit_array(w)))
19          else
20             for all i s.t. bit_array(w)[i]=1 do
21                M.addEdge(associated_node(i), w)
22 return M
```

nodes, that is, nodes around which the map is built. The data structure `reach` (e.g., a hash table) is used to track the nodes reached at each iteration of the external `for` loop (see line 1).

The algorithm starts (STEP 0) by picking one node $v \in V_M$ (line 1) and by setting all the nodes in V_Γ as not reached from v (line 3). The algorithm then performs a variant of BFS starting from v (i.e., V_1 in Fig. 9). Fig. 9 also shows in STEP 1 and STEP 2 the first two iterations of the BFS (lines 7–16). The first node in the queue Q is extracted (i.e., V_1); then for all nodes reachable from this node, the algorithm checks if they have already been reached (lines 10–11) and changes their corresponding values in the `reach` data structure. Moreover, if these nodes belong to the set V_M (nodes in the map) then the corresponding edge is added to the output map and the number of reached nodes is incremented (STEP 2 in Fig. 9). The variable `num_reached` serves the purpose to stop the BFS from the current node if all the distinguished nodes have already been added to the map. If nodes encountered during the BFS are not distinguished (they do not belong to V_M) then they are inserted in the queue Q. Note that it is crucial to consider all the nodes in V_M (line 1) although they may have already been added to the map. Consider, for instance, the edge e between V_{11} and V_5. If V_{11} is not considered because it has been already added to the map in STEP 2, this edge would not be present in the map. Fig. 9 STEP n shows the final result of the algorithm.

3.4. Algebra of maps

In this section, we research algebraic properties of maps and define operations over them. The following theorem shows the properties of a family of maps.

Theorem 3.11. *Let $\Gamma = (V_\Gamma, E_\Gamma)$ be a Web Region and $\mathcal{M}(\Gamma)$ the set of all good maps over Γ. Let $M_i = (V_{M_i}, E_{M_i}) \in \mathcal{M}(\Gamma)$ be the good map obtained by choosing the set of distinguished nodes $V_{M_i} \subseteq V_\Gamma$. Then:*

1. *The binary relation $\sqsubseteq$ over $\mathcal{M}(\Gamma)$, defined by $M_1 \sqsubseteq M_2$ iff $V_{M_1} \subseteq V_{M_2}$, is a partial order on $\mathcal{M}(\Gamma)$.*
2. *The order $\sqsubseteq$ induces a Boolean algebra $(\mathcal{M}(\Gamma), \sqcup, \sqcap, \Gamma, \emptyset)$, where: $M_1 \sqcup M_2$ is the unique good map of Γ over $V_{M_1} \cup V_{M_2}$; $M_1 \sqcap M_2$ is the unique good map of Γ over $V_{M_1} \cap V_{M_2}$.*
3. *There is an isomorphism of Boolean algebras from $(\mathcal{P}(V), \cup, \cap, V, \emptyset)$ to $(\mathcal{M}(\Gamma), \sqcup, \sqcap, \Gamma, \emptyset)$, given by $N \mapsto M_N$ (the unique good map of N over Γ).*

Proof. *1.* The relation is clearly reflexive and transitive. For antisymmetry: if $M_1 \sqsubseteq M_2$ and $M_1 \sqsubseteq M_2$, then $V_{M_1} = V_{M_2}$. Moreover it holds that $E_{M_1} \subseteq E^*_{M_2}$ and $E_{M_2} \subseteq E^*_{M_1}$, where $E^*_{M_i}$ is the set of edges of the transitive closure of M_i. From Lemma 3.4 and the goodness of both maps, it follows: $x \to y$ in E_{M_1} iff $x \twoheadrightarrow_N y$ in Γ iff $x \to y$ in E_{M_2}. Items *2.* and *3.* are long, but straightforward, and they check for definitions of Boolean algebra and isomorphism between algebras. □

Having well defined operations over maps enables one to obtain new maps from other maps. This enables the combining of maps as described in Example 2.3. The crucial question is if the re-computation of a map can be (partially) avoided. The results show this possibility.

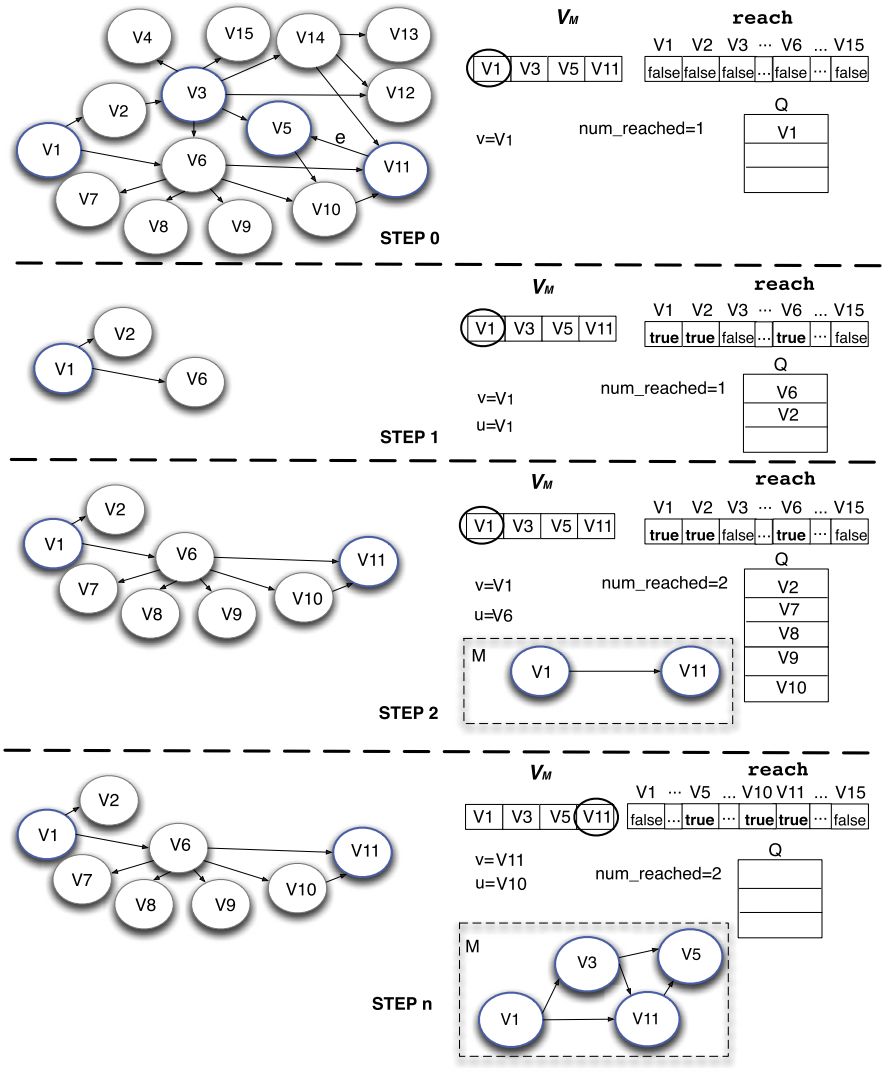

Fig. 9. Constructing the map of the region Γ.

Proposition 3.12. *Let* $M_i = (V_{M_i}, E_{M_i})$ $i = 1, 2$ *be good maps over* Γ *and let* M_N^Γ *denote the good map of* Γ *over the set of nodes* N. *Then:*

1. $M_1 \sqcap M_2 = M^{M_1}_{V_{M_1} \cap V_{M_2}} \cup M^{M_2}_{V_{M_1} \cap V_{M_2}}$
2. $E_{M_1 \sqcup M_2} \subseteq E_{M_1} \cup E_{M_2} \cup$
   $\cup\{(x, y) \in E_\Gamma : x \in V_{M_1}, y \in V_{M_2}\} \cup \{(y, x) \in E_\Gamma : x \in V_{M_1}, y \in V_{M_2}\} \cup$
   $\cup\{x \twoheadrightarrow_{V_{M_1} \cup V_{M_2}} y, x \in V_{M_1}, y \in V_{M_2}\} \cup \{y \twoheadrightarrow_{V_{M_1} \cup V_{M_2}} x, x \in V_{M_1}, y \in V_{M_2}\}$

Proof. [PART 1]. Let $x, y \in V_{M_1} \cap V_{M_2}$ and $x \to y$ in $M_1 \sqcap M_2$ we show that either $x \to y$ in $M^{M_1}_{V_{M_1} \cap V_{M_2}}$ or $x \to y$ in $M^{M_2}_{V_{M_1} \cap V_{M_2}}$ holds. For $x \to y$ to be in $M_1 \sqcap M_2$ than $x \twoheadrightarrow_{V_{M_1} \cap V_{M_2}} y$ in Γ. We can follow in three cases: *i)* the edge $x \to y$ is in M_1 or M_2, and thus belongs to the good map over M_1 or M_2; *ii)* the path $x \twoheadrightarrow_{V_{M_1} \cap V_{M_2}} y$ in Γ passes through at least one node z such that $z \in V_{M_1}$ and $z \notin V_{M_2}$, and thus $x \to y$ is in $M^{M_1}_{V_{M_1} \cap V_{M_2}}$; *iii)* the path $x \twoheadrightarrow_{V_{M_1} \cap V_{M_2}} y$ in Γ passes through at least one node z such that $z \in V_{M_2}$ and $z \notin V_{M_1}$, and thus $x \to y$ is in $M^{M_2}_{V_{M_1} \cap V_{M_2}}$. Now, we show that if $x, y \in V_{M_1} \cap V_{M_2}$ and $x \to y$ in $M^{M_1}_{V_{M_1} \cap V_{M_2}}$ or $x \to y$ in $M^{M_2}_{V_{M_1} \cap V_{M_2}}$ than $x \to y$ in $M_1 \sqcap M_2$ holds. It is enough to note that each edge $x \to y$ in M_1 implies that there exists a path $x \twoheadrightarrow_{V_{M_1}} y$ in Γ and thus also $x \twoheadrightarrow_{V_{M_1} \cap V_{M_2}} y$ in Γ holds and we can conclude that if $x, y \in V_{M_1} \cap V_{M_2}$ than $x \to y$ in $M_1 \sqcap M_2$ must hold. The same applies to edges in M_2.

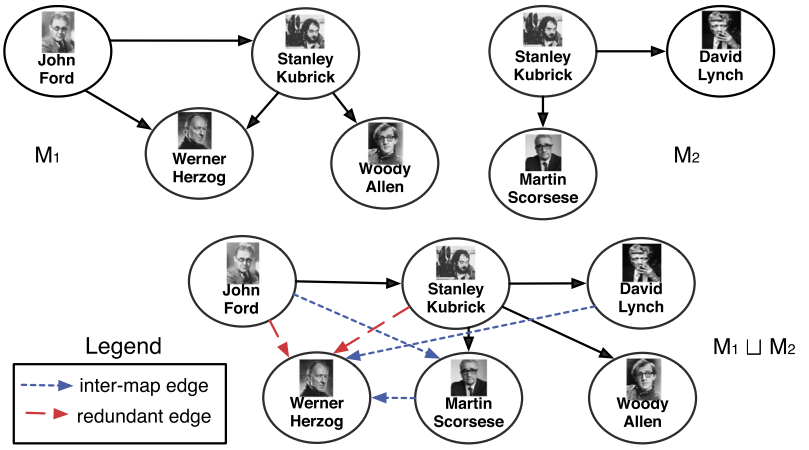

Fig. 10. Map of the union of M_1 and M_2.

Then either: *i)* the edge $x \to y$ is in M_1 or M_2, and thus belongs to the good map over M_1 or M_2 or *ii)* the path $x \twoheadrightarrow_{V_{M_1} \cap V_{M_2}} y$ in Γ can be represented as a sequence of 1's (nodes in M_1), 2's (nodes in M_2) and 0's (nodes neither in M_1 nor M_2). If the sequence contains a 1, then $x \twoheadrightarrow y$ in M_1: consider all 1's in the sequence; between each pair of them there is a path, either direct (thus in M_1) or indirect (using only nodes 2 and 0 and thus also in M_1). Thus, the edge $x \to y$ is in $M_{V_{M_1} \cap V_{M_2}}^{M_1}$. A similarly reasoning holds for terms involving M_2.

[PART 2]. If $x \to y$ in $E_{M_1 \sqcup M_2}$, then either there is a path in E_{M_1} or in E_{M_2} or if x, y were in different V_{M_i}'s, then the path belongs to one of the four sets described. □

Corollary 3.13. *Let M_1 and M_2 be two maps over Γ. Then:*

1. *The map $M_1 \sqcap M_2$ can be computed by using information available in the maps M_1, M_2 only, and in time $\mathcal{O}(|V_{M_1} \cap V_{M_2}| \times (|V_{M_1}| + |E_{M_1}| + |V_{M_2}| + |E_{M_2}|)$.*
2. *The superset of the edges of $M_1 \sqcup M_2$ as discussed in Proposition 3.12 cannot be computed more efficiently than computing the good map over $V_{M_1} \cup V_{M_2}$ from scratch.*

Proof. [PART 1]. It follows from Theorem 3.10. [PART 2]. It is necessary to check for the existence of paths from nodes in V_{M_1} to nodes in V_{M_2} (and the other way around) in Γ. This check has a cost no lower than computing the good map of Γ on $V_{M_1} \cup V_{M_2}$. Note that edges $x \to y$ in M_1 could come from a path $x \to z \to y$ with $z \in V_{M_2}$ and after the union could become redundant; but this redundancy cannot be checked only with information in M_1 and M_2. □

An example of the algebra is reported in Fig. 10 where the map of Syd's favorite directors and that of Joe's favorite directors are united (see Example 2.3). The edge from J. Ford to W. Herzog in M_1 is redundant with respect to the definition of good map of the region in Fig. 5. Indeed, the only path between the nodes is via M. Scorsese. A similar reasoning holds for the edge between S. Kubrick and W. Herzog. Note that some edges (e.g., M. Scorsese to W. Herzog) link nodes originally belonging to different maps.

4. Declarative specification of graph regions

In this section, we discuss the problem of data selection. We tackle data selection by using navigational languages that allow the declarative specification of regions of graphs and keep information about reachability among nodes of interest.

In the remainder of this section, we focus our attention on directed graphs with both node and edge labels that are useful to declaratively specify and retrieve Web Regions via navigational languages. Node labels play the role of node identifiers, both in the graph and in its regions. Note that even if edges in the graph have different labels, since regions capture reachability information, edges in the region assume the intrinsic meaning of *is reachable from* independently from their actual label (e.g., an edge from A to B in the region with whatever label means that B is reachable from A in the graph). In other words, edge labels do not play any role in the building of maps once the region is available. The need to capture reachability among nodes in a graph that is crucial for building regions and maps, is codified in the following general problem:

Problem 1. Given a graph $G = (V_G, E_G)$ and a set of nodes $N \subseteq V_G$, construct a subgraph (a region) $R = (V', E')$ of G such that $N \subseteq V'$.

Variants of this problem have been extensively studied. Faloutsos el al. [17] addressed the following problem: given an edge-weighted undirected graph, two vertices s, t, an integer k, find a connected subgraph H of size k containing s, t that maximizes a given goodness function. Other approaches based on data mining techniques have been proposed to discover groups of people (e.g., [18]), measure the reliability of networks (e.g., [19]), simplify networks (e.g., [20]) or obtain informative connections (e.g., [21]). Yet other approaches compute spanning trees [22], Steiner trees [23], and more complex constructions, like k-cliques in Social Network Analysis [24].

Generally speaking, the starting point of these approaches is to apply some specific technique to the whole graph G, which is assumed to be *locally* available. These approaches are not suitable for our purpose since the Web structure is discovered on the fly and the whole Web graph is not known a priori. Our departure point is different: we leverage logical/algebraic methods to organize and specify in a formal, machine-understandable and reusable way, subgraphs (i.e., regions) and maps of the Web.

4.1. A general graph navigational language

We want to solve the problem of how to isolate specific regions of a graph G in order to work with them. This is particularly important when dealing with the Web graph, which is not locally available or when one is interested in only a specific region of the graph (e.g., in bibliographic networks). The following question arises: how do we formally specify and obtain regions of graphs? Graph navigational languages partially address this issue, leading us to the following variant of Problem 1.

Problem 2. Given a directed graph $G = (V_G, E_G)$ and a set of nodes $N \subseteq V_G$ obtained from the evaluation of a query Q in a navigational language, construct a subgraph (a region) $\Gamma = (V_\Gamma, E_\Gamma)$ of G such that $N \subseteq V_\Gamma$ and Γ capture information relevant to Q.

Abstractly defined, a navigational language $\mathcal{L}$ over a graph $G = (V_G, E_G)$ is a set of functions ("queries") of the form $V_G \rightarrow \text{subgraphs}(G) \times \mathcal{P}(V_G)$ that assign to each node v a subgraph (the visited nodes and edges) plus a set of distinguished nodes (the resources selected). Many navigational languages (e.g., XPath [25], nSPARQL [14]) consider labeled graphs and are able to find pairs of nodes connected by a sequence of edge labels matching a pattern (or navigational expression) expressed via regular expressions over the alphabet of edge labels. Navigation is only referenced in the semantics as the means to get the resulting set of pairs of nodes. The relevant aspect of navigational languages is that their classical semantics is not enough for our goal; we need information about reachability in order to deal with regions and maps. To address this problem, we introduce the general navigational language $\mathcal{L}$, by taking inspiration from the NautiLOD language that was initially proposed with a semantics returning nodes [15] and then equipped with a semantics returning graph fragments in addition to sets of nodes [9,26–28]. The idea is that information on the navigational paths, as well as the resources selected give us the piece of semantic information to build regions. Indeed, the paths traversed when finding nodes satisfying an expression are relevant to enrich information given by the nodes themselves. The syntax of $\mathcal{L}$ is:

$$\texttt{path} ::= \texttt{label} \mid \texttt{path[test]} \mid \texttt{(path)*} \mid \texttt{(path|path)} \mid \texttt{path/path}$$

The core of $\mathcal{L}$ is standard regular expressions built upon edge labels and boolean tests used to drive the navigation. In Section 5.1 we instantiate the language in the Web of Linked Data. Here, edge labels will concretely represent RDF predicates and tests will be of different types, for instance, ASK SPARQL queries (as in NautiLOD [9,15,27]) and/or nested regular expressions (e.g., as in nSPARQL [14]). The reader can refer to [27] for a comprehensive discussion about Web navigational languages.

4.2. Graph regions obtained via navigational languages

A graph region Γ specified by a navigational expression is a tuple $\langle (V_\Gamma, E_\Gamma), N \rangle$ where (V_Γ, E_Γ) is a standard directed graph and $N \subseteq V_\Gamma$ is a set of distinguished nodes. To access the elements of Γ we use the notation $\Gamma.x$ where $x \in \{V_\Gamma, E_\Gamma, N\}$. In particular, $\Gamma.N$ is the set containing the starting node from which the evaluation of an expression begins and the set of ending nodes, that is, the expression endpoints. A graph region Γ captures the specification given via a navigational expression e starting from a single node of the graph where $\Gamma.E_\Gamma$ and $\Gamma.V_\Gamma$ are edges and nodes navigated during the evaluation of e. The need to start from a node (or a set of nodes) is motivated by the fact that our framework is meant to be used on the Web, and thus, it is infeasible to start the evaluation of a navigational expression from all nodes of the Web graph (assuming one can enumerate all of them). Moreover, this also mimics the process of manual navigation on the Web where one starts from a single (or a small set of) Web page.

The semantics of the general language $\mathcal{L}$ is reported in Table 1, where $\mathcal{D}(\texttt{u})$ is the set of outgoing edges of $\texttt{u}$. The evaluation of an expression $\texttt{path}$ over the Graph G is performed starting from a seed node $\texttt{u}$ according to the rules R1–R16 (here the graph G is omitted for conciseness). The evaluation returns an ordered pair, that is, the directed graph and the set of distinguished nodes that correspond to the graph region Γ. The main module is the function $E_{\mathcal{V}}$ (rule R1) that represents the starting point of the evaluation. The other modules are the following:

Table 1
The semantics of the general Language $\mathcal{L}$ returning regions. The rules reflect all the possible forms of syntactic expressions that can be given according to the $\mathcal{L}$ syntax.

R1	$E_V[\![\texttt{path}]\!](\mathrm{u}) = \langle (V_n[\![\texttt{path}]\!](\mathrm{u}), V_e[\![\texttt{path}]\!](\mathrm{u})), \{\mathrm{u}\} \cup V_r[\![\texttt{path}]\!](\mathrm{u})\rangle$
R2	$V_n[\![\texttt{label}]\!](\mathrm{u}) = \{\mathrm{u}, \mathrm{v} \mid (\mathrm{u}, \texttt{label}, \mathrm{v}) \in \mathcal{D}(\mathrm{u})\}$
R3	$V_n[\![\texttt{path[test]}]\!](\mathrm{u}) = V_n[\![\texttt{path}]\!](\mathrm{u})$
R4	$V_n[\![(\texttt{path})^*]\!](\mathrm{u}) = \{\mathrm{u}\} \cup (\bigcup_{i=1}^{\infty} V_n[\![(\texttt{path}_i)]\!](\mathrm{u}) \mid \texttt{path}_1 = \texttt{path} \wedge \texttt{path}_i = \texttt{path}_{i-1}/\texttt{path})$
R5	$V_n[\![\texttt{path}_1/\texttt{path}_2]\!](\mathrm{u}) = V_n[\![\texttt{path}_1]\!](\mathrm{u}) \cup \bigcup \mathrm{v} \in V_r[\![\texttt{path}_1]\!](\mathrm{u}) V_n[\![\texttt{path}_2]\!](\mathrm{v})$
R6	$V_n[\![\texttt{path}_1\vert\texttt{path}_2]\!](\mathrm{u}) = V_n[\![\texttt{path}_1]\!](\mathrm{u}) \cup V_n[\![\texttt{path}_2]\!](\mathrm{u})$
R7	$V_e[\![\texttt{label}]\!](\mathrm{u}) = \{(\mathrm{u}, \mathrm{v}) \mid (\mathrm{u}, \texttt{label}, \mathrm{v}) \in \mathcal{D}(\mathrm{u})\}$
R8	$V_e[\![\texttt{path[test]}]\!](\mathrm{u}) = V_e[\![\texttt{path}]\!](\mathrm{u})$
R9	$V_e[\![(\texttt{path})^*]\!](\mathrm{u}) = \bigcup_{i=1}^{\infty} V_e[\![(\texttt{path}_i)]\!](\mathrm{u}) \mid \texttt{path}_1 = \texttt{path} \wedge \texttt{path}_i = \texttt{path}_{i-1}/\texttt{path}$
R10	$V_e[\![\texttt{path}_1/\texttt{path}_2]\!](\mathrm{u}) = V_e[\![\texttt{path}_1]\!](\mathrm{u}) \cup \bigcup_{\mathrm{v} \in V_r[\![\texttt{path}_1]\!](\mathrm{u})} V_e[\![\texttt{path}_2]\!](\mathrm{v})$
R11	$V_e[\![\texttt{path}_1\vert\texttt{path}_2]\!](\mathrm{u}) = V_e[\![\texttt{path}_1]\!](\mathrm{u}) \cup V_e[\![\texttt{path}_2]\!](\mathrm{u})$
R12	$V_r[\![\texttt{label}]\!](\mathrm{u}) = \{\mathrm{v} \mid (\mathrm{u}, \texttt{label}, \mathrm{v}) \in \mathcal{D}(\mathrm{u})\}$
R13	$V_r[\![\texttt{path[test]}]\!](\mathrm{u}) = \{\mathrm{v} \mid \mathrm{v} \in V_r[\![\texttt{path}]\!](\mathrm{u}) \wedge \texttt{Evaluate}(\texttt{test}, \mathrm{v}) = \texttt{true}\}$
R14	$V_r[\![(\texttt{path})^*]\!](\mathrm{u}) = \{\mathrm{u}\} \cup (\bigcup_{i=1}^{\infty} V_r[\![(\texttt{path}_i)]\!](\mathrm{u}) \mid \texttt{path}_1 = \texttt{path} \wedge \texttt{path}_i = \texttt{path}_{i-1}/\texttt{path})$
R15	$V_r[\![\texttt{path}_1/\texttt{path}_2]\!](\mathrm{u}) = \bigcup \mathrm{v} \in V_r[\![\texttt{path}_1]\!](\mathrm{u}) V_r[\![\texttt{path}_2]\!](\mathrm{v})$
R16	$V_r[\![\texttt{path}_1\vert\texttt{path}_2]\!](\mathrm{u}) = V_r[\![\texttt{path}_1]\!](\mathrm{u}) \cup V_r[\![\texttt{path}_2]\!](\mathrm{u})$

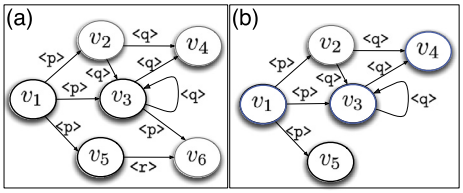

Fig. 11. An example of (a) graph and a region (b). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

- $V_n[\![\texttt{path}]\!](\mathrm{u})$: This function takes as input a `path` (i.e., an expression defined according to the $\mathcal{L}$ syntax) and a URI `u`. The function gives as output the set of all nodes visited during the evaluation of `path`;
- $V_e[\![\texttt{path}]\!](\mathrm{u})$: This function takes as input a `path`, which is evaluated starting from the URI `u`, and gives as output the set of edges traversed during the evaluation of `path`.
- $V_r[\![\texttt{path}]\!](\mathrm{u})$: This function takes as input a `path`, which is evaluated starting from the URI `u`, and gives as output the set of nodes reachable via sequences of edges satisfying `path` (i.e., result nodes).

Note that the classic semantics of XPath, nSPARQL, NautiLOD and other navigational languages returning the endpoints can only be captured via this structure (i.e., $\langle (N, \emptyset), N \rangle$). The $\mathcal{L}$ language is parametric in the type of tests that can be handled by the function `Evaluate` (rule R13).

An example of evaluation returning a graph region. We now provide an example of the evaluation of the expression $\texttt{path} = \texttt{p/q}$ in the language $\mathcal{L}$ returning a graph region. Consider the graph in Fig. 11 (a) and v_1 as a starting point for the evaluation.

Let Γ be the graph region that we want to build. The starting point for the evaluation of `path` is Rule R1 applied starting from node v_1. According to R1, v_1 is immediately added to the set $\Gamma.N$. Then, the graph (V_Γ, E_Γ) is built by calling the functions V_n and V_e, respectively. Similarly the other distinguished nodes are obtained via the function V_r. Since these functions are very similar to one another we will discuss in detail only the application of V_r. The first rule applied is R15 that makes usage of rule R12 on the label `p`. Rule R12 looks for edges labeled as `p` originating from v_1. This causes the traversing of the edges $\langle v_1, \mathrm{p}, v_2\rangle$, $\langle v_1, \mathrm{p}, v_3\rangle$ and $\langle v_1, \mathrm{p}, v_5\rangle$ (that are added to $\Gamma.E_\Gamma$ via rule R7; moreover, nodes v_2, v_3 and v_5 are also added to $\Gamma.V_\Gamma$ via rule R2). Then, rule R15 calls again rule R12 starting from each of v_2, v_3 and v_5 looking for edges labeled as `q`. Rule R12 considers the edges $\langle v_2, \mathrm{q}, v_3\rangle$, $\langle v_2, c, v_4\rangle$, $\langle v_3, \mathrm{q}, v_3\rangle$ and $\langle v_3, \mathrm{q}, v_4\rangle$ (again note that the corresponding nodes and edges are added to $\Gamma.V_\Gamma$ via rule R2 and to $\Gamma.E_\Gamma$ via rule R7). Nodes v_3 and v_4 are added to $\Gamma.N$ by R15. The resulting Web Region is shown in Fig. 11(b), where distinguished nodes (i.e., nodes in $\Gamma.N$ are shown in blue).

4.3. Extracting regions from the web of linked data graph

This section deals with the extraction of regions from the Web of Linked Data graph. There is a huge amount of RDF data shared and interlinked on the Web. This is evolving the traditional Web of Documents into a Web of (Linked) Data

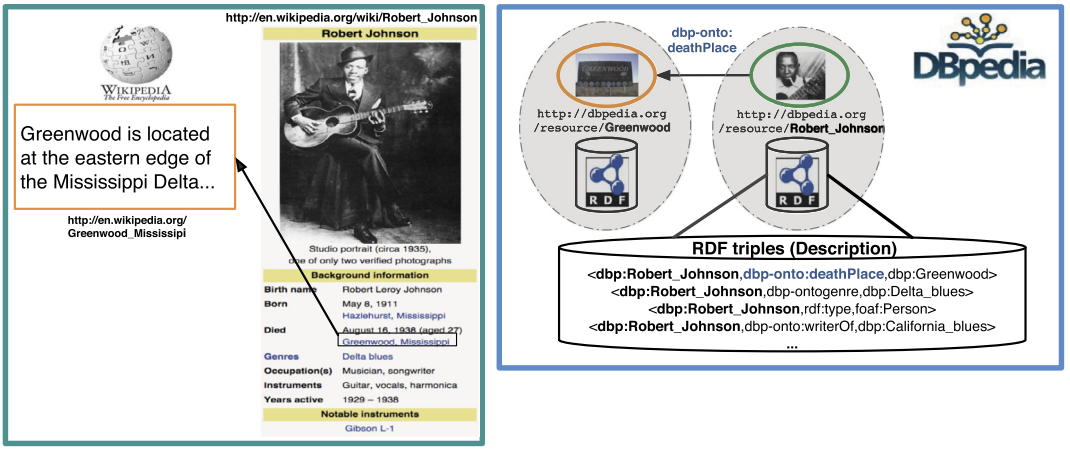

Fig. 12. Web of documents vs. Web of (Linked) Data.

(WoD) [12]. Here, 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., places) and abstract concepts (e.g., 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. However, semantic links and descriptions are now expressed by using a common data format, that is, RDF. Fig. 12 offers a glimpse into the ideas underlying the WoD. The left part shows an excerpt of two HTML pages: one about R. Johnson and the other about the city where he passed away. The link between the two pages is a syntactic link. In the right part we have the same pieces of information represented in RDF. In this case the link (i.e., an RDF predicate) is given a meaningful label, and information about R. Johnson is given in the form of RDF triples.

Theorem 4.1. *The Web Region* Γ *of the WoD graph corresponding to an expression* e *in the language* $\mathcal{L}$ *can be extracted with no overhead, in term of the time required, with respect to the cost of evaluating* e.

To build Γ it is enough to store each RDF triple navigated when evaluating e. Note that in this context, the function $\mathcal{D}(\mathtt{u})$ used in Table 1 simulate the URI dereferencing operation, which returns an RDF graph of triples about $\mathtt{u}$.

5. Implementation and examples

The framework to build maps of the Web of Linked Data has been implemented in Java in the MaGe tool. Expressions in the language $\mathcal{L}$ to specify regions, regions and maps are represented in RDF and can be shared, exchanged and reused. Note that maps of general graphs (as discussed in Section 3) do not have edge labels, although an edge has the implicit semantics of "reachable"; an edge between nodes A and B means that B can be reached from A by not passing through distinguished nodes. In the RDF representation of a map the edge semantics is made explicit via a URI (i.e., "reachable"). The MaGe tool, the files used in the experiments and the results are available at the MaGe Web site.[8] Fig. 13 shows the high-level architecture of the tool.

The architecture comprises two main modules, the *selection* module and the *abstraction* module. The first module is responsible for the implementation of the language $\mathcal{L}$. In particular, given a seed URI and an expression in the language $\mathcal{L}$, this module retrieves a Web Region. The retrieval occurs as follows. The *Interpreter* receives the input, checks the syntax and initializes both the *Execution Manager* (with the *seed* URI) and the *Automaton builder*, which generates the automaton associated with the expression, which will be used to drive the evaluation of the expression on the Web. The *Execution Manager* controls the flow of the execution and passes the URIs to be dereferenced to the *Network Manager*, which performs the dereferencing of URIs via HTTP GET calls. The sets of RDF triples obtained are converted into Jena[9] models by the *RDF Manager*. The *Link Extractor* takes in input the automaton and the model and selects a subset of outgoing links (to be possibly traversed at the next step of the navigation) according to the current state of the automaton. All the outgoing links (that are RDF triples) selected are added to the region Γ being built. Moreover, this set of links is given to the *Execution Manager*, which starts the cycle over.

The execution ends when there are no more URIs to be dereferenced. The *abstraction* module implements the map framework discussed in Section 3 and given a Web Region, it applies the algorithms presented in Section 3.2 to construct maps. The decoupling between selection and abstraction enables their use also separately.

[8] http://mapsforweb.wordpress.com.
[9] http://apache.jena.org.

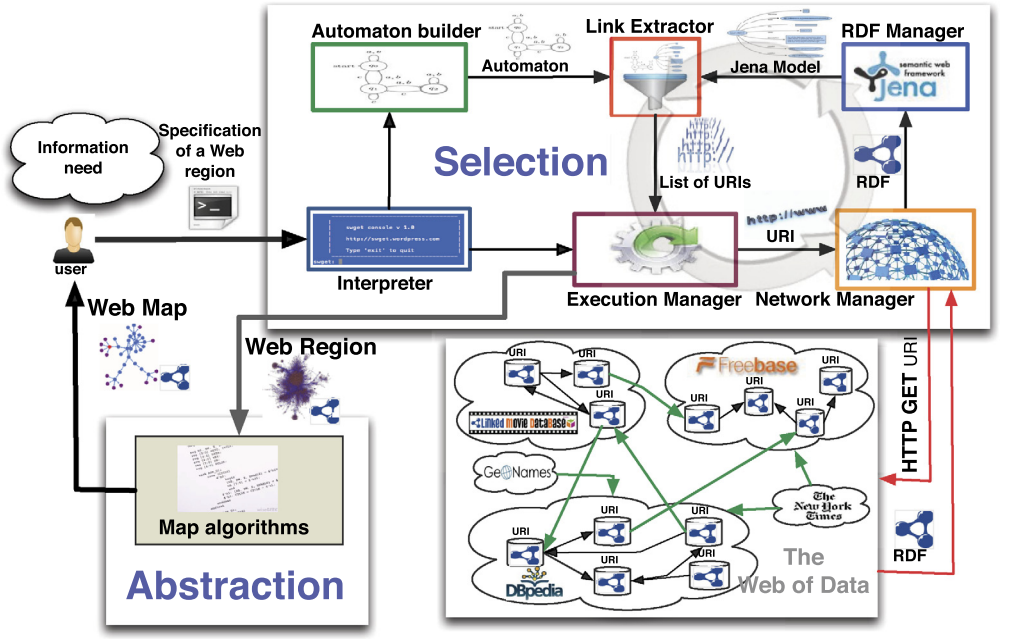

Fig. 13. The architecture of MAGE.

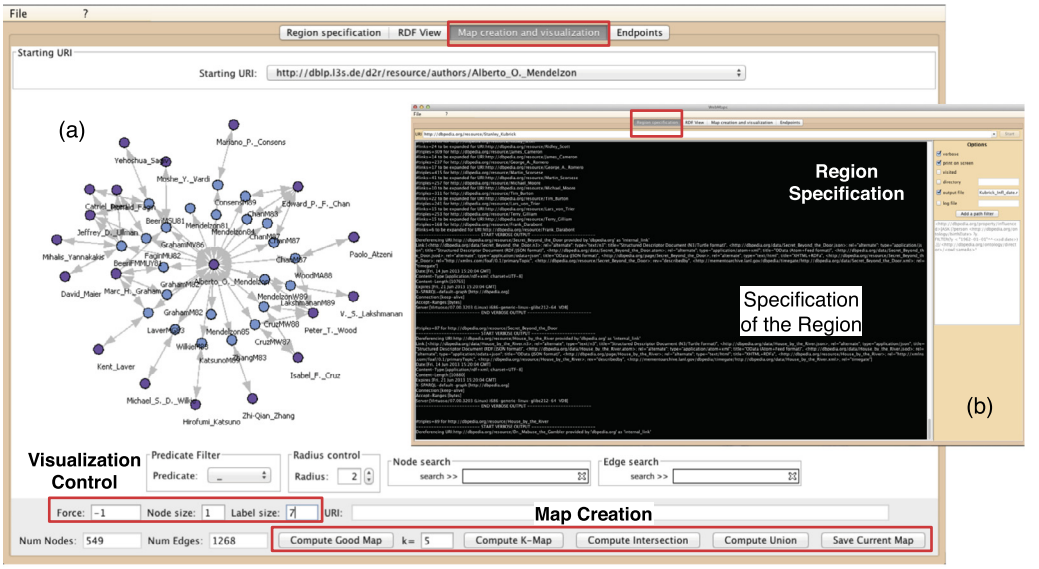

Fig. 14. The GUI of the MAGE system.

MAGE is also endowed wit a GUI (shown in Fig. 14). It includes four main tabs. The first one is used to specify the region via an expression in $\mathcal{L}$ (shown in Fig. 14(b)). The second and fourth display the region retrieved in RDF and the expression endpoints. The third tab deals with the creation of maps and their visualization (shown in Fig. 14(a)). The bottom-right section enables one to compute the good map of the region and the k-maps. It is possible to save the map in RDF and use the algebra of maps. The bottom-left part enables controlling the visualization of the region/map.

5.1. Toward building web maps

Fig. 15 summarizes all the elements discussed so far to build maps of Web Regions. In particular, regions from the Web of Linked Data are obtained via formal specifications in the language $\mathcal{L}$ (see the first arrow in Fig. 15). A region is the portion of the Web of Linked Data that is navigated during the evaluation of a navigational expression and it is represented as a directed graph (as discussed in Section 4). Regions are then abstracted by using the framework introduced in Section 3 to

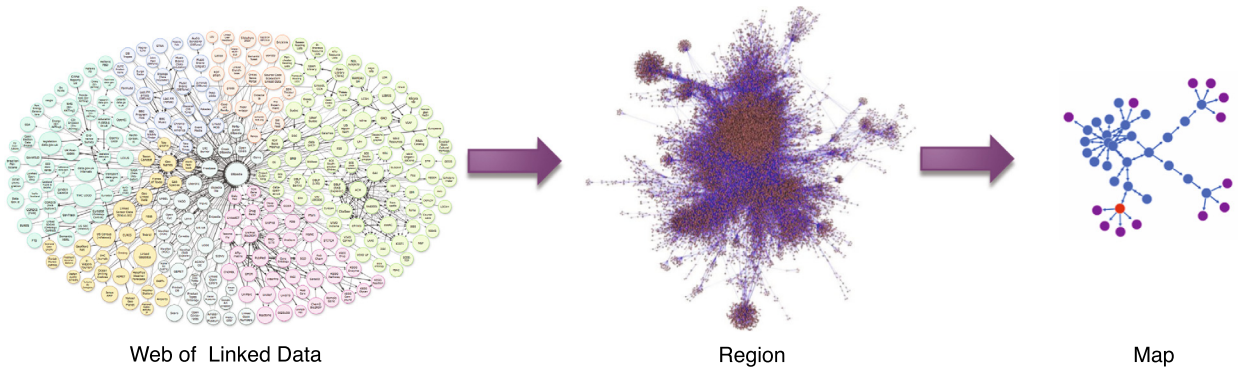

Fig. 15. Obtaining maps of the Web.

obtain a compact representation of reachability information among a set of nodes that have been marked as 'distinguished' in the region (see the second arrow in Fig. 15).

Overall, the properties and efficiency arguments of the whole map framework are summarized in the next result that follows from Theorem 4.1 and Theorem 4.1.

Proposition 5.1. *Given an expression e in the language $\mathcal{L}$ that specifies a region $\Gamma = \langle (V_\Gamma, E_\Gamma), N \rangle$ of the WoD graph:*

The good map $M = (N, E_M)$ can be built in time $\mathcal{O}(|N| \times (|V_\Gamma \setminus N| + |E_\Gamma|)) + eval(e) + C_{tests}$, where $eval(e)$ is the cost of evaluating the expression e and C_{tests} the cost of evaluating the tests in e.

Proof. Given the expression e, the corresponding Web Region can be built in time $eval(e) + C_{tests}$ (see Theorem 4.1 and Theorem 3.10). Having the region, the good map can be computed in time $\mathcal{O}(|N| \times (|V_\Gamma \setminus N| + |E_\Gamma|))$ where $|V_\Gamma|$ and $|E_\Gamma|$ are the number of nodes and edges of the region and N is the set of distinguished nodes selected by e. □

Note that the space required to evaluate an $\mathcal{L}$ expression and build the (k-)map of the corresponding region is proportional to the time, since only relevant nodes are visited. Indeed, the semantics of Γ is such that $x \twoheadrightarrow y$ in Γ if and only if there is a navigational path from x to y satisfying a subexpression of e. As a consequence, since edges in M correspond to paths, in Γ each $x \rightarrow y$ in M means that there is a navigational path from x to y satisfying a subexpression of e.

5.2. Real world examples

We discuss some examples of Web Maps in two contexts: influence networks and bibliographic networks. These examples motivate the introduction of our Web Map framework as they cannot be (at least not easily) simulated by using current navigational languages and map making tools. This is for two main reasons: i) typically navigational languages lack the ability to return subgraphs connecting nodes of interest; ii) current approaches able to retrieve subgraphs do not provide any abstraction mechanism. We restate the following terminology: *(i)* the *starting* node is the URI from which the navigational expression is evaluated; *(ii)* the *ending* nodes are the nodes (URIs) satisfying the expression. The function f used to compute k-maps considers the degree centrality of nodes.

Map of influence networks. An influence network is a graph where nodes are different kinds of persons and edges represent influence relations. We leverage information taken from `dbpedia.org` where the notion of influence is captured via the property `dbp:influenced` defined in the DBpedia ontology. We consider two different regions: a region built starting from Tim Berners Lee (TBL) and another starting from Stanley Kubrick (SK).

Example 5.2 *(Maps of influence networks and algebra).* Specify two regions that contain people that have influenced or have been influenced up to distance 6 by TBL and SK, respectively. The ending nodes in the regions must be scientists. Compute maps and combine them using the algebra.

The regions can be specified with the following expression by considering the URIs of TBL (`dbpedia:Tim_Berners-Lee`) and SK (`dbpedia:Stanley_Kubrick`) in DBPedia as starting nodes, respectively:

```
dbp:influenced<1-6>[test]
```

where `test= ASK {?ctx rdf:type dbpedia:Scientist.}`

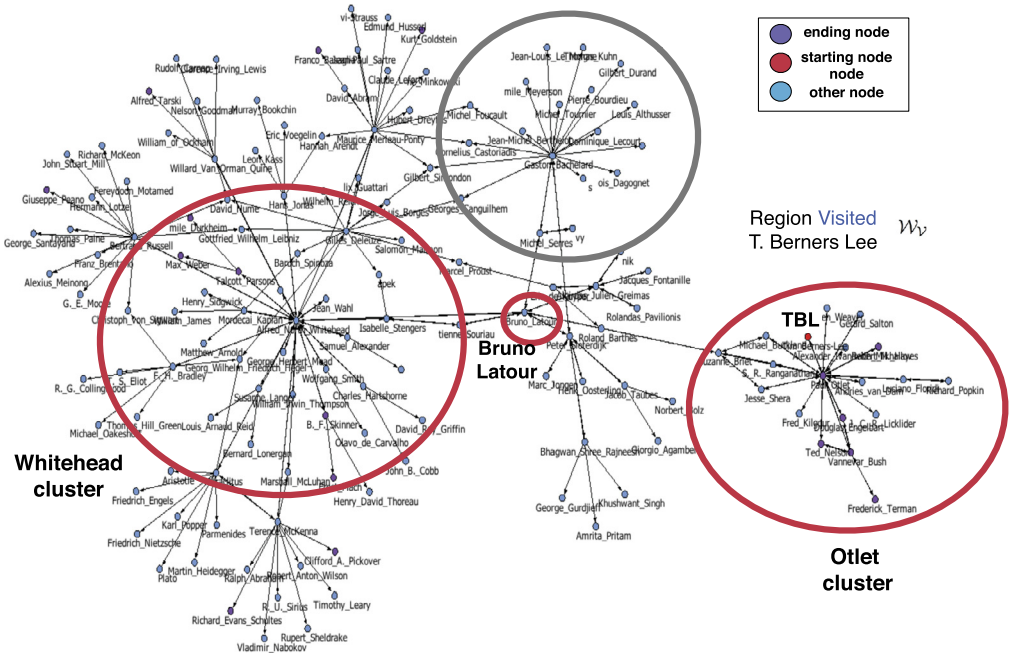

Fig. 16. Region obtained for TBL. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

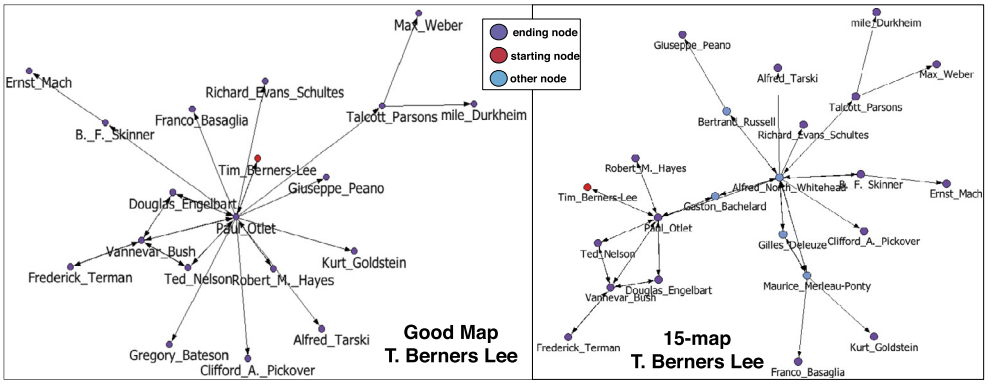

Fig. 17. Good map and 15-map for TBL.

The notation `<1-6>` is a shortcut to indicate up to 6 concatenations of the predicate `dbp:influenced` and the special variable `?ctx` in the test indicates the current node reached during the evaluation of the expression.

Influences of TBL. Fig. 16 reports the region obtained by evaluating the expression starting from TBL. This region contains 149 nodes and 236 edges. Recall that the specification of the region stated that only scientists should be considered as the ending nodes. As it can be observed, there are two main influence clusters; the first around Paul Otlet and the second one around Alfred North Whitehead. Observe that while Otlet is an ending node (i.e., he is a scientist and thus it is reported in violet), Whitehead is not (i.e., he is not a scientist and thus it is reported in light blue); however, via Whitehead other scientists (e.g., Peano) have been reached. Note also that Bruno Latour is the bridge between the two clusters and he is not a scientist. Moreover, the region also contains an influence-(quasi)clique of scientists that includes Otlet, Nelson, Bush and Engelbar.

The good map and the 15-map for TBL are reported in Fig. 17. Recall that the aim of the good map (18 nodes; 43 edges) is to represent in a concise way reachability information among distinguished nodes (scientists in this case) in the region.

This abstract representation tells us, for instance, that there exists an influence from TBL to G. Peano passing via P. Outlet. If one is interested in the whole path (also including non-scientists) it can be retrieved by looking to the region (see Fig. 16), obtaining TBL→P. Outlet→S. Briet→B. Latour→A.N. Whitehead→B. Russel→G. Peano.

To include more information in the good map one can use k-maps. The notion of k-map is meant to provide a mechanism to construct multi-granular maps of Web Regions. By computing the 15-map of the region (23 nodes; 43 edges), it can be

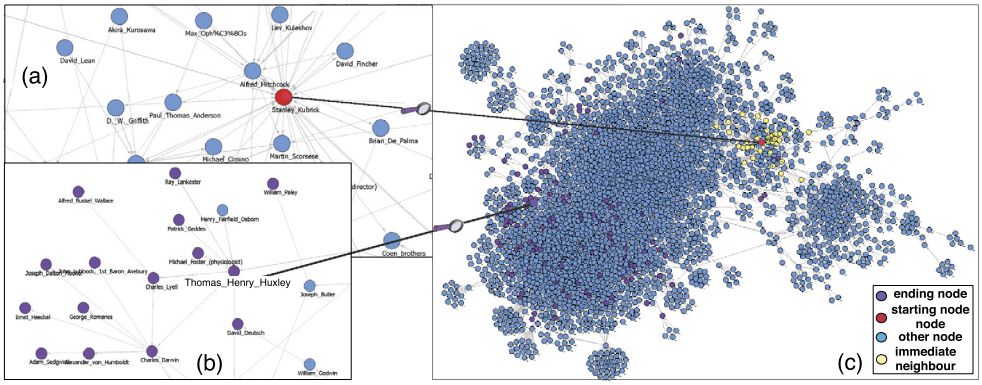

Fig. 18. Region obtained for SK (c) and some zooms (a)–(b). This type of visualization (with small labels and bigger nodes) has been obtained from the *Map Creation and Visualization* tab of MAGE.

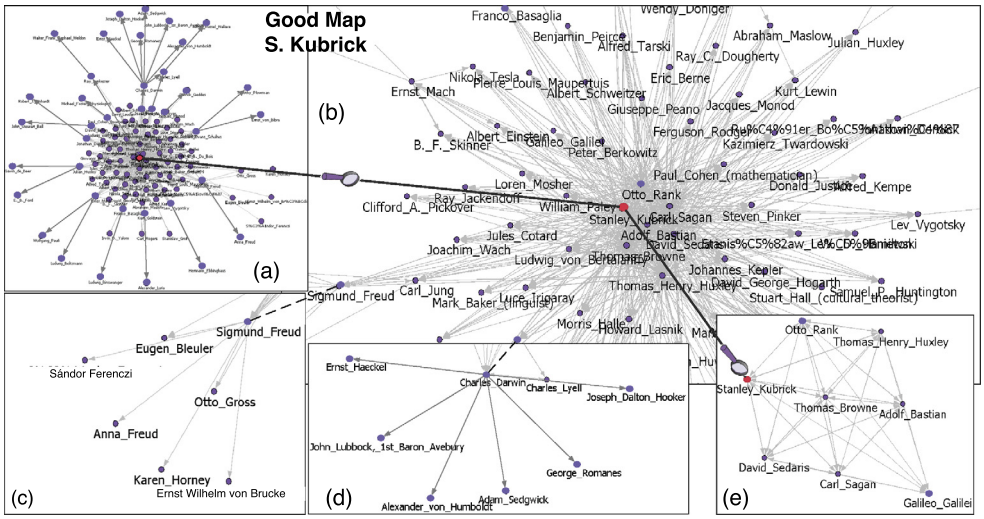

Fig. 19. Good map for SK (a) and some zooms (b)–(e).

noted that some of the non-ending nodes in the path from TBL to G. Peano (i.e., A.N. Whitehead and B. Russel) are present in the 15-map meaning that their degree centrality in the region is greater or equal to 15.

Influences of SK. Fig. 18 reports the region associated with the influence network of SK. The region contains 2,981 nodes and 7,893 edges. It is very difficult to identify the ending nodes and more importantly reachability among them and with the starting node. To address this issue we computed the good map associated with this region (109 nodes; 2,629 edges), which is reported in Fig. 19. The abstraction provided by the good map enables identifying the influence path between SK and C. Segan. We also zoomed in on this influence path by computing the 60-map (not shown here) of the region (120 nodes; 3,627 edges). This showed that there is an influence path between SK and C. Segan (ending node) including the non-ending nodes Burroughs, Cameron and Whedon.

Algebra of maps. Let M_1 be the 60-map and M_2 be the 8-map obtained from the influence regions of TBL and SK respectively, Fig. 20 shows examples of the algebra of maps. In particular, Fig. 20 (a) shows the union and Fig. 20 (b) the intersection. The map of the union aids in putting together information from the two maps. Generally speaking, it discovers possible additional influence relations between pairs of nodes that are not present in the same map. In this specific example, there is no path between SK and TBL in either map or regions. However, the union discovers the connection between TBL and SK (i.e., TBL→P. Outlet→S. Briet→A.N. Withehead←SK). The intersection has been computed as the union of the good maps of M_1 and M_2 over $V_{M_1} \cap V_{M_2}$ (see Section 3.4).

The advantage of using the algebra is to avoid having to compute the good map from scratch and obtain it without looking at the regions. As an example, in the intersection we have the nodes G. Peano and A. Tarski, which means that they both belong to the influence networks of SK and TBL. As traditional maps help users orient themselves in large and unknown spaces, the possibility to specify regions and automatically build and combine maps opens up new possibilities for users to make sense of the Web.

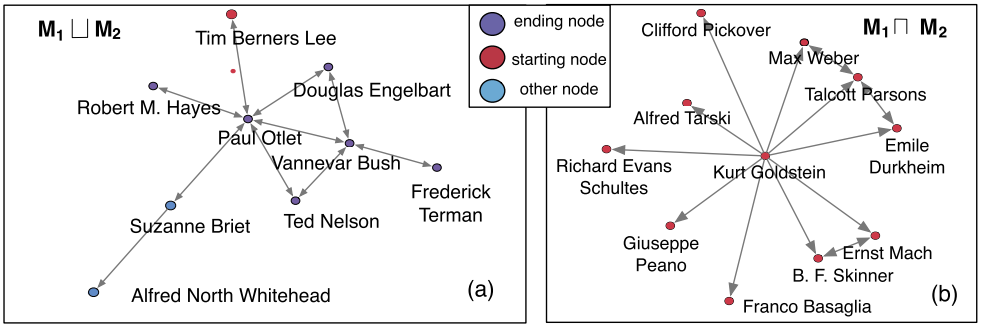

Fig. 20. Algebra of maps between the 60-map of SK (M_1) and the 8-map of TBL (M_2).

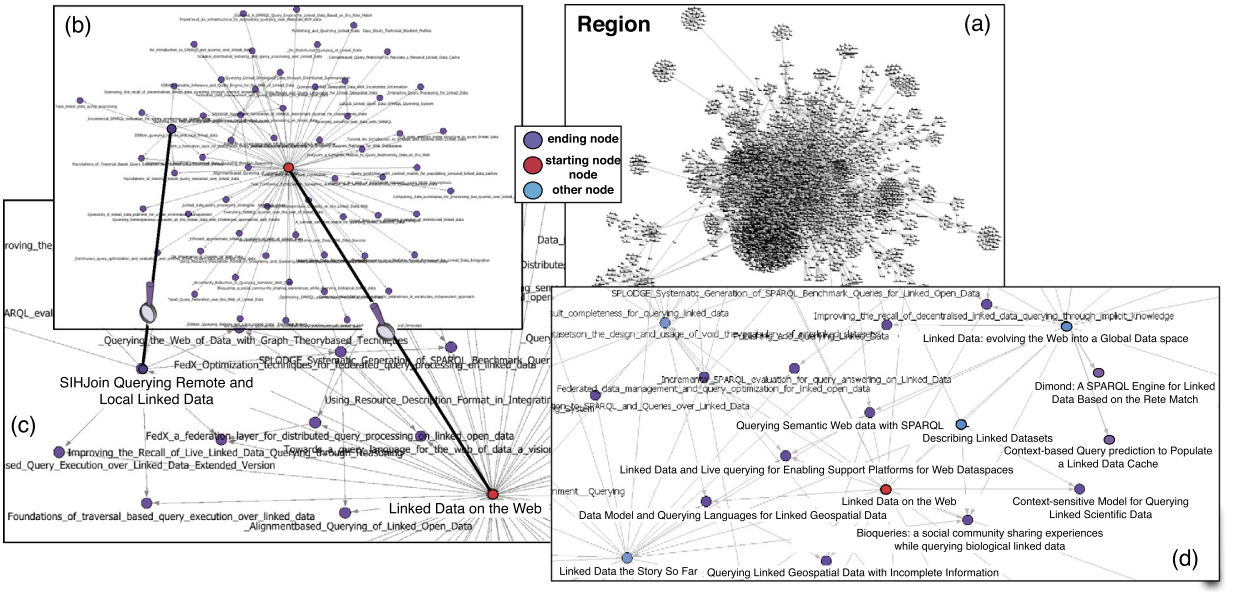

Fig. 21. Citation region.

Bibliographic networks. Current approaches for building maps of bibliographic networks mainly focus on techniques such as clustering and information content (e.g., [29]). Tools like DBLP, Google Scholar and Microsoft Academic Search provide Web interfaces to access bibliographic data. One can manually "navigate" the networks of papers and citations and manually mark those of interest. However, connections among these papers are lost. The drawback of these approaches is that they lack features of graph query languages such as mechanisms to specify a particular region of interest in the network to use for building maps. Consider the following example:

Example 5.3 *(Map of citations).* We want to specify a region containing papers up to distance three from the paper "Linked Data on the Web". The ending nodes must be only papers whose title contains the set of keywords "Linked", "Data" and "Quer*" or the set "Web", "Data" and "Quer*". We want then to build different maps of this region.

The region can be specified with the following expression (e_1) in $\mathcal{L}$ considering the URI of the paper "Linked Data on the Web" in Google Scholar as starting node:

```
citedBy<1-3>[test]
```

In the expression, the notation `<1-3>` is a shorthand for the concatenation of (up to) three `citedBy` while

```
test= title.contains(Linked, Data, Quer*) | title.contains(Web, Data, Quer*).
```

Fig. 21 shows the regions obtained (3,462 nodes and 4,804 edges). Note that a region is not a map and does not provide any level of abstraction. Without abstracting the region, it is difficult to identify the ending nodes, connections between

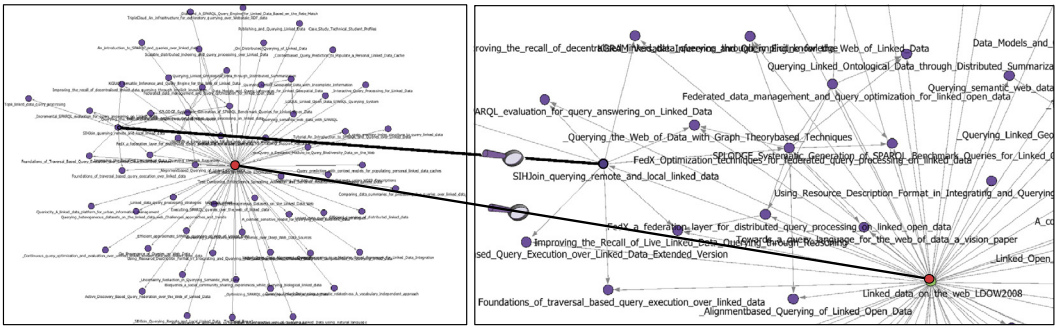

Fig. 22. The good map for Example 5.3.

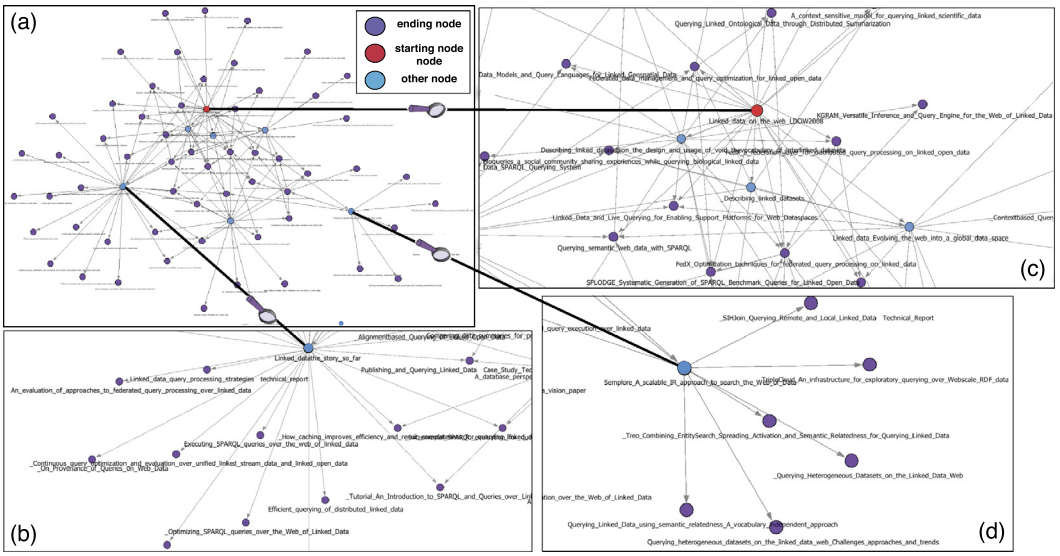

Fig. 23. 7-map for Example 5.3.

them and with the starting node. Fig. 22 shows the good map for Example 5.3. In a different manner from the region, the good map promotes quick insights among the ending nodes of the expression. For instance, one can see that the papers "SIHJoin: Querying Remote and Local Linked Data" and "Foundations of Traversal based Query Execution over Linked Data" are connected both to the seed paper (as one would expect) and also to each other. Fig. 23 shows the 7-map of the region. The map gives further details for the path between the starting node and the node "A Context Sensitive Model for Querying Linked Scientific Data" by including the node "Describing Linked Data Sets". Fig. 24 shows how the number of nodes and edges (y axis) varies with k; the more we zoom in (decreasing k) the more nodes and edges are added to the map.

6. Related work

The development of our map framework shares some characteristics with other research areas. We discuss each of these research areas separately.

6.1. Summarization/compression of graphs

The problem of graph summarization has been deeply studied. Approaches like [30,31] focus on the problem of discovering properties of large social networks (e.g., scale-free property, small world effect). Although their main aim is produce compact representations, these summaries are not graphs. A closely related strand of research is graph compression (e.g., [32,33]). The main difference with the present work is that the goal of compression is to reduce the storage space needed to represent the original graph; in other words, minimize the number of bits needed to represent the graph. Moreover, multiple edge types are not considered. Graph partitioning methods (e.g., [34,35]) are useful in detecting, for instance, communities. However, properties of nodes are largely ignored in the analysis.

Faloutsos et al. [17] focus on the task of finding the best connected subgraph of a graph structure containing two distinguished nodes. Adibi et al. [18] face the problem of finding groups in a large network while Hintsanen [19] studies

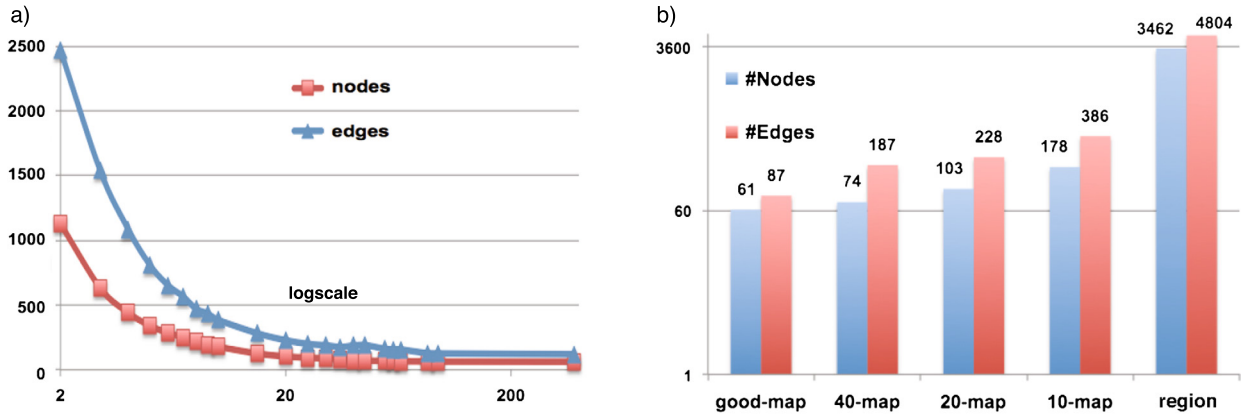

Fig. 24. Some statistics about Example 5.3.

the problem of finding the most reliable subgraph in a graph subject to edge and node failures. More general approaches seek to discover informative subgraphs [21] while others with simplifying general network structures [20] or determining specific substructures such as the minimum spanning tree [36]. The problem of simplification is also relevant in the area of *complex networks* (e.g., [29]) and Semantic Graphs. The work by Barthélemy et al. [37] aims to establish the relevance of links, via schema information and metrics defined in the area of complex networks, for detecting relationships.

All the aforementioned approaches have the following aim: given a network structure N, determine a function F in order to find a simplified structure N^s satisfying some requirements. F usually exploits some techniques such as data mining or information content. We underline the following main differences with the present work. First, the function F aims at simplifying the *whole* network structure. It is not possible to specify a region of interest to simplify. Second, the simplified network N^s is represented in an ad-hoc format and it is not meant to be exchanged, reused and combined with other structures (e.g., via an algebra). Third, these approaches do not work on the Web; this is because they assume the availability of the whole graph while our approach dynamically discovers the region of interest in the graph.

6.2. Specification of regions

This line of related work concerns languages for the declarative specification of nodes in a graph or sub-graphs (e.g., XPath [25] for XML and other approaches for general graphs [38]). In the specific case of RDF, there have been some proposals extending the SPARQL query language with navigational features (e.g., [39,14,40,41], SPARQL 1.1). Other languages (e.g., NautiLOD [15]) have been designed to enable semantic navigation in the Web of Linked Data. All these approaches in general do not face the problem of summarizing regions. They focus on specifying sets of nodes that are identified by evaluating an expression over a graph. No information about their connections is provided and then they are not suitable for building maps.

There is some available work on specifying/reconstructing structures such as Consens et al. [42] who face the problem of reconstructing the portion of an XML document that contributes to answering an XPath query. Anyanwu et al. [43] define a subgraph extraction language called SPARQL2L for RDF. Our approach differs from these in an important respect: they focus on identifying a partial slice of data of a particular interest but do not attempt to map, in the sense to represent "all" data in a suitable mathematical form. Our approach is meant to define maps and the aspect of *abstraction* and manipulation via formally defined operators is crucial. The notion of summary has been exploited in Harth et al. [44]; although upon summarization, this work does not focus on defining techniques to specify maps, foster their exchange, reuse and composition. There is a crucial difference between summaries, indexes and maps: a summary is a brief statement of the main points of something while an index is an alphabetical list with references to the place where some piece of information can be found.

6.3. Construction of maps

Research in graph visualization aims at designing layout methods to improve the visualization of large graphs (see [45] for a survey). The goal of this paper is to produce maps that can be interpreted also by machines and with formal provable relations. Hence, the problem of visualization is orthogonal; however, our approach can be combined with visualization methods to facilitate the user interpretation of maps. In the remainder of this section we consider approaches that focus on providing some form of "map" of the Web.

Dodge [2] in his *Atlas of the Cyberspace*, provides a comprehensive overview of visual representations of *digital landscapes* on the Web. Boder et al. [46] studied the topology of the Web while Bizer et al. [47] studied the topology of the Web of Data. These studies are meant to provide a deeper comprehension of (large) interconnected information spaces [48] or knowledge domains [49]. A recent information visualization paradigm used to summarize information is that of *metro maps* (e.g., [6,7]). Other strands of research related to ours are (visual) navigational histories, site maps and bookmarks.

Parunak [50] and McEneaney [51] studied the abstract problem of user navigation. Doemel [5] described a system called WebMap, which visualizes the topology of the visited region of the Web graph by calculating the spanning tree. An approach enabling the creation of concept maps is described in Gaines and Shaw [52]. Navigation enhanced by geospatial information is studied in McCurley [53]. As for site maps, Li et al. [54] researched the problem of building multi-granular site maps. Finally, bookmarking consists in marking and sharing (e.g., with Delicious) URIs for the purpose of future reuse.

The crucial difference with the present work is that these approaches are designed for human usage and are mainly oriented to visualization; hence they do not allow automatic processing, composition and reuse, thus hindering the automation of the process of creating, exchanging, combining and interpreting maps. Second, they do not include formal/provable relations of reachability between the URLs chosen; and formal provable relations between the map and the region it represents, thus obstructing the generation of formal deductions from them. Third, there is no possibility to declaratively specify regions of interest to build maps.

Another strand of research related to ours is that of topic maps, mind maps and concept maps [55]. As an example, topic maps [56] are a new standard for describing knowledge structures and associating them with information resources. The main difference with the present work is that these objects are manually constructed; they are not meant to provide a concise representation of a specification with formal provable properties.

7. Concluding remarks

Due to limitations of human I/O capabilities, the management of information at a Web scale calls for automatic mechanisms and thus machine-processable information. In this paper we have shown that maps, key devices in helping human navigation in information spaces, are meaningful on the Web space, and furthermore, their generation and manipulation can be semi-automated. Our proposal offers a refreshing perspective to the idea of *Associative Trails* defined by V. Bush in his proposal of the Memex [57]. Our framework enables a user to automatically build *"a trail of his interest in the maze of materials available"* on the Web. We think that the formal models presented here are a starting point to further develop the discipline of cartography on the Web. We have shown the feasibility of such formal models by implementing them over the current Web infrastructure (i.e., data and languages today available). As a proof of concept in this paper, we used the $\mathcal{L}$ language and the notion of k-map, and we are aware that the framework developed would benefit from using other languages and notions of relevant maps. In particular, the maps we discussed are represented as unlabeled graphs. This is because maps can be labeled according to different strategies, for instance, path length or summary of labels in the represented paths. In this version, we remain "neutral" with respect to the labeling to highlight the very general idea of map. But clearly a map with labeled edges is also interesting. In particular, each edge in the map abstracts a set of paths between distinguished nodes in the region. Since edges in a path (a sequence of RDF predicates) can have different labels, assigning a single label to each edge in the map does not reflect the meaning of the paths they represent. To overcome this problem, each edge of the map could be labeled with a regular expression summarizing all the paths connecting such nodes in the region. This way also information about the sequences of edge labels between distinguished nodes in the region would remain in the map.

References

[1] A.H. Robinson, J. Morrison, O.C. Muehrcke, A.J. Kimerling, S.C. Guptill, Elements of Cartography, Wiley, 1995.
[2] M. Dodge, R. Kitchin, Atlas of Cyberspace, Addison–Wesley, Great Britain, 2001.
[3] M. Rosvall, C. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. USA 105 (4) (2008) 1118–1123.
[4] A.M. Ledbetter, E. Griffin, G.G. Sparks, Forecasting "friends forever": a longitudinal investigation of sustained closeness between best friends, Pers. Relatsh. 14 (2) (2007) 343–350.
[5] P. Doemel, WebMap: a graphical hypertext navigation tool, Comput. Netw. ISDN Syst. 28 (1) (1995) 85–97.
[6] E. Sandvad, K. Grønbæ k, L. Sloth, J. Knudsen, A metro map metaphor for guided tours on the web: the webvise guided tour system, in: WWW, 2001, pp. 326–333.
[7] D. Shahaf, C. Guestrin, E. Horvitz, Trains of thought: generating information maps, in: WWW, 2012, pp. 899–908.
[8] G. Pirrò, Explaining and suggesting relatedness in knowledge graphs, in: ISWC, 2015, pp. 622–639.
[9] V. Fionda, C. Gutierrez, G. Pirrò, Knowledge maps of web graphs, in: KR, 2014.
[10] V. Fionda, C. Gutierrez, G. Pirrò, The map generator tool, in: ISWC (Posters & Demos), 2014, pp. 81–84.
[11] J.L. Borges, On the exactitude of science, in: Jorge Luis Borges, Collected Fictions, 1998, p. 325, Trans. H. Hurley.
[12] C. Bizer, T. Heath, T. Berners-Lee, Linked data – the story so far, Int. J. Semantic Web Inf. Syst. 5 (3) (2009) 1–22.
[13] S. Harris, A. Seaborne, SPARQL 1.1 query language [online], 2010.
[14] J. Pérez, M. Arenas, C. Gutierrez, NSPARQL: a navigational language for RDF, J. Web Semant. 8 (4) (2010) 255–270.
[15] V. Fionda, C. Gutierrez, G. Pirrò, Semantic navigation on the web of data: specification of routes, web fragments and actions, in: WWW, 2012, pp. 281–290.
[16] T. Heath, C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Morgan & Claypool, 2011.
[17] C. Faloutsos, K. McCurley, A. Tomkins, Fast discovery of connection subgraphs, in: KDD, 2004, pp. 118–127.
[18] J. Adibi, H. Chalupsky, E. Melz, A. Valente, et al., The KOJAK group finder: connecting the dots via integrated knowledge-based and statistical reasoning, in: AAAI, 2004, pp. 800–807.
[19] P. Hintsanen, The most reliable subgraph problem, in: PKDD 2007, 2007, pp. 471–478.
[20] F. Zhou, S. Malher, H. Toivonen, Network simplification with minimal loss of connectivity, in: ICDM, 2010, pp. 659–668.
[21] C. Ramakrishnan, W. Milnor, M. Perry, A. Sheth, Discovering informative connection subgraphs in multi-relational graphs, ACM SIGKDD Explor. Newsl. 7 (2) (2005) 56–63.
[22] P. Macdonald, E. Almaas, A.L. Barabási, Minimum spanning trees of weighted scale-free networks, Europhys. Lett. 72 (2) (2005) 308.

[23] G. Kasneci, M. Ramanath, M. Sozio, F.M. Suchanek, G. Weikum, STAR: Steiner-tree approximation in relationship graphs, in: ICDE, 2009, pp. 868–879.
[24] R.A. Hanneman, M. Riddle, Introduction to Social Network Methods, University of California, 2005.
[25] W.W.W. Consortium, XML Path language (XPath) recommendation, 1999.
[26] V. Fionda, C. Gutierrez, G. Pirrò, Extracting relevant subgraphs from graph navigation, in: ISWC (Posters & Demos), vol. 914, 2012.
[27] V. Fionda, G. Pirrò, C. Gutierrez, Nautilod: a formal language for the web of data graph, ACM Trans. Web 9 (1) (2015) 5:1–5:43.
[28] V. Fionda, C. Gutierrez, G. Pirrò, The swget portal: navigating and acting on the web of linked data, J. Web Semant. 26 (2014) 29–35.
[29] M. Rosvall, D. Axelsson, C. Bergstrom, The map equation, Eur. Phys. J. Spec. Top. 178 (1) (2009) 13–23.
[30] D. Chakrabarti, C. Faloutsos, Graph mining: laws, generators, and algorithms, ACM Comput. Surv. 38 (1) (2006) 2.
[31] M. Newman, The structure and function of complex networks, SIAM Rev. 45 (2) (2003) 167–256.
[32] D.K. Blandford, G.E. Blelloch, I.A. Kash, Compact representations of separable graphs, in: ACM–SIAM, 2003, pp. 679–688.
[33] X. He, M.-Y. Kao, H.-I. Lu, A fast general methodology for information-theoretically optimal encodings of graphs, SIAM J. Comput. 30 (3) (2000) 838–846.
[34] M.E. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E, Stat. Nonlinear Soft Matter Phys. 69 (2) (2004) 026113.
[35] X. Xu, N. Yuruk, Z. Feng, T.A. Schweiger, Scan: a structural clustering algorithm for networks, in: SIGKDD, 2007, pp. 824–833.
[36] H.N. Gabow, Z. Galil, T.H. Spencer, R.E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, Combinatorica 6 (2) (1986) 109–122.
[37] T.E.-R. Marc Barthélemy, Edmond Chow, Knowledge representation issues in semantic graphs for relationship detection, in: AAAI Spring Symposium: AI Technologies for Homeland Security, 2005, pp. 91–98.
[38] R. Angles, C. Gutierrez, Survey of graph database models, ACM Comput. Surv. 40 (1) (2008) 1–39.
[39] F. Alkhateeb, J.-F. Baget, J. Euzenat, Extending SPARQL with regular expression patterns (for querying RDF), J. Web Semant. 7 (2) (2009) 57–73.
[40] H. Zauner, B. Linse, T. Furche, F. Bry, A RPL through RDF: expressive navigation in RDF graphs, in: RR, 2010, pp. 251–257.
[41] V. Fionda, G. Pirrò, M. Consens, Extended property paths: writing more SPARQL queries in a succinct way, in: AAAI, 2015, pp. 102–108.
[42] M.P. Consens, J.W.S. Liu, F. Rizzolo, XPlainer: visual explanations of XPath queries, in: ICDE, 2007, pp. 636–645.
[43] K. Anyanwu, A. Maduko, A. Sheth, SPARQL2l: towards support for subgraph extraction queries in RDF databases, in: WWW, 2007, pp. 797–806.
[44] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K. Sattler, J. Umbrich, Data summaries for on-demand queries over linked data, in: WWW, 2010, pp. 411–420.
[45] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice Hall PTR, 1998.
[46] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener, Graph structure in the web, Comput. Netw. 33 (1–6) (2000) 309–320.
[47] C. Bizer, P. Mendes, A. Jentzsch, Topology of the web of data, in: Semantic Search over the Web, 2012, pp. 3–29.
[48] O. Turetken, R. Sharda, Visualization of web spaces: state of the art and future directions, ACM SIGMIS Database 38 (3) (2007) 51–81.
[49] K. Borner, C. Chen, K. Boyack, Visualizing knowledge domains, Annu. Rev. Inf. Sci. Technol. 37 (1) (2005) 179–255.
[50] H.V.D. Parunak, Hypermedia topologies and user navigation, in: Hypertext, 1989, pp. 43–50.
[51] J. McEneaney, Visualizing and assessing navigation in hypertext, in: Hypertext, 1999, pp. 61–70.
[52] B. Gaines, M. Shaw, WebMap: concept mapping on the web, in: WWW, 1995.
[53] K. McCurley, Geospatial mapping and navigation of the web, in: WWW, 2001, pp. 221–229.
[54] W. Li, N. Ayan, O. Kolak, Q. Vu, H. Takano, H. Shimamura, Constructing multi-granular and topic-focused web site maps, in: WWW, 2001, pp. 343–354.
[55] A. Okada, S. Shum, J. Buckingham, T. Sherbone, Knowledge Cartography: Software Tools and Mapping Techniques, Springer, 2008.
[56] S.R. Newcomb, A perspective on the quest for global knowledge interchange, in: XML Topic Maps: Creating and Using Topic Maps for the Web, 2002, pp. 31–50.
[57] V. Bush, As we may think, Atl. Mon. (1945).