UNIVERSITÁ DI PISA
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE

INTERACTIVE VISUALIZATION OF SEMANTIC WEB DATA

DOCTORAL THESIS

Author
Fabio Valsecchi

Tutor (s)
Dr. Andrea Marchetti and Prof. Marco Avvenuti

Reviewer (s)
Dr. Aba-Sah Dadzie and Prof. Fabio Vitali

The Coordinator of the PhD Program
Prof. Marco Luise

Pisa, November 2016

XXIX

Acknowledgements

WORKING on the visualization of abstract data using the cartographic metaphor has been a very interesting and stimulating experience. The scientific and artistic nature of the topics and activities I performed during these three years allowed me to unleash my creativity. For this reason but also for his guidance, availability for the discussions we had and the brainstorming sessions we carried out, a special thank goes to Matteo Abrate.

I would like also to thank my supervisors, Andrea Marchetti for giving me the possibility of working in his research group and increase my skills and knowledge, and Marco Avvenuti for his support during the whole Ph.D. program.

Summary

LINKED DATA represent a huge source of information and knowledge both inside and outside the Semantic Web research field. However, a broad and extensive adoption of this technology is still prevented by the difficulties that users find when approaching these large and complexly structured sources of data. Typically, even the experts of this field are disoriented in understanding both the main structure and the details characterizing how these data have been modelled. Filling this initial gap means performing very meticulous, costly and time-consuming analysis by using a specific query language.

A possible way for solving the problem is adopting other approaches that are more centered on the use of visual representations, leveraging the human visual perceptual channel. Differently from the ordinary methods widely investigated in the literature, we propose an approach based on Information Visualization techniques and cartographic principles. Large and complex data are exactly the kind of information Cartography has been dealing with for centuries. This capability in representing data can be further augmented by the interactive mechanisms that can be implemented using modern computer-based solutions.

Navigating geographic spaces and effectively observing their information is a task humans are quite used to. The approach presented in this thesis produces abstract maps resembling the traditional geographic ones and thus allows users to reuse their cognitive skills and prior knowledge in reading maps for the navigation of abstract map-like visualizations of Linked Data sets. In order to produce these particular visualizations, a specific process called spatialization has been employed to assign data a range of spatial attributes, such as size, shape and position on a two-dimensional surface. The specific spatialization we used is based on space-filling curves, fractal curves having the feature of entirely filling the space. In particular, by exploiting the fractal nature of these curves, a novel technique for properly expressing data and efficiently spatialize them in a scalable way has been devised.

Sommario

I Linked Data rappresentano un'enorme fonte di informazione e conoscenza sia all'interno che all'esterno del settore di ricerca del Semantic Web. Un'ampia adozione di questa tecnologia risulta purtroppo essere ancora frenata dalle difficoltà che gli utenti incontrano nell'approcciare questi insiemi di dati caratterizzati da un elevato numero di elementi e da una struttura molto complessa. Solitamente anche gli esperti di questo ambito sono inizialmente disorientati nel capire quale sia esattamente loro la struttura e i dettagli riguardanti la loro modellazione. Per colmare questo divario è necessario effettuare un'analisi molto meticolosa, onerosa e che richiede molto tempo utlizzando linguaggi di interrogazione specifici.

Una possibile soluzione consiste nell'adottare un approccio diverso, più focalizzato sull'utilizzo di rappresentazioni visuali che possano sfruttare il canale di percezione visivo umano. Diversamente dai metodi comuni molto utilizzati negli studi presenti in letteratura, proponiamo un approccio basato su tecniche di Information Visualization e su principi cartografici. Grandi e complesse quantità di dati sono esattamente il tipo di informazioni che la cartografia gestisce ormai da secoli. Questa capacità può essere ulteriormente aumentata da meccanismi interattivi implementabili da moderne soluzioni software.

Navigare spazi geografici ed osservarne particolari informazioni è un processo che le persone sono ormai abituate a fare. L'approccio presentato in questa tesi produce delle mappe astratte che somigliano a quelle geografiche tradizionali e quindi permettono agli utenti di riusare le loro capacità cognitive e le pregresse abilità nella lettura di mappe. Per poter generare queste visualizzazioni deve essere impiegato un processo specifico di spazializzazione in modo da assegnare ai dati un insieme di attributi spaziali come dimensione, forma e posizione su un piano bidimensionale. La spazializzazione utilizzata è basata su un tipo particolare di curve frattali chiamate space-filling curve, caratterizzate dalla proprietà di coprire interamente lo spazio a disposizione. In particolare, sfruttando la composizione frattale di queste curve, è stata ideata una nuova tecnica per esprimere i dati e spazializzarli in maniera efficiente e scalabile.

Summary of Ph.D. achievements

During the three years of the Ph.D. I performed my research activities at the Institute of Informatics and Telematics (IIT) of the National Research Council (CNR) of Pisa, Italy. More specifically, I worked in Web Applications for the Future Internet (WAFI) group.

Research activities

- *Semantic Analysis of Information for Anti-evasion* (ASIA) is a regional project focused on the collection and analysis of information retrieved from the Web about people and commercial activities. My activities concerned the design and development of a system for gathering and analyzing data for finally enriching the profiles of people and commercial activities in order to highlight anomalies due to tax evasion.

- *OpeNER* is a FP7 European project with the purpose of providing the industries of the tourism sector with different technologies for the recognition of entities of interest such as hotels and restaurants, their classification and the sentiment analysis of their corresponding reviews. My activities consisted in designing and developing heuristic algorithms for performing and evaluating the entity linking of the entities of interest the project collected.

- *Smart Campus* is a national project consisting in the construction of innovative systems for improving the working quality and optimizing the energetic resources. My activities focused on the design and development of web application consisting in a interactive map allowing the research of places and the navigation of the CNR campus in Pisa.

- *Clavius on the Web* is a project founded by the Registro.it aiming to study and add value to the correspondence of Christophorus Clavius. My activities mainly consisted in the management of the Linked Data of the project and the design and development of a tool for the annotation of manuscripts using Semantic Web technologies.

- *DoGi* is a database containing bibliographic references of articles published on Italian juridical journals. Linked Data DoGi is an interdepartmental project consisting on the translation of a relational database into a new one relying on Semantic Web technologies. My activities focused on the design and development of a system for publishing, querying and visually presenting Linked Data.

International Conferences/Workshops with Peer Review

1. Valsecchi, F., Ronchetti, M. (2014, August). Spacetime: a Two Dimensions Search and Visualisation Engine Based on Linked Data. International Conference on Advances in Semantic Processing. (Vol. x, pp. 8).
2. Valsecchi, F., Abrate, M., Bacciu, C., Tesconi, M., Marchetti, A. (2015, May). DBpedia atlas: Mapping the uncharted lands of linked data. World Wide Web Conference, Linked Data on the Web Workshop. (Vol. 1409, pp. 25).
3. Valsecchi, F., Abrate, M., Bacciu, C., Tesconi, M., Marchetti, A. (2015, October). Linked Data Maps: Providing a Visual Entry Point for the Exploration of Datasets. International Semantic Web Conference, Intelligent Exploration of Semantic Data. (Vol. 1472, pp. 1).
4. Valsecchi, F., Abrate, M., Bacciu, C., Piccini, S., Marchetti, A. (2016, October). Text Encoder and Annotator: an all-in-one editor for transcribing and annotating manuscripts with RDF. European Semantic Web Conference, Semantic Web for Scientific Heritage. (Vol. 1595, pp. 53).

Contents

CHAPTER 1

Introduction

INSIDE and outside the Computer Science community, 1991 will be always remembered as the birth of the World Wide Web (WWW). It was when its inventor, Tim Berners-Lee, a young researcher working at CERN, the European Particle Physics Laboratory in Geneva, published on-line the first web page[1] of the history of the WWW. The project had an immediate effect, so viral that the number of available Web servers grew to 500 on the end of 1993 and widely exploded to 10.000 in late 1994[2]. At the time, Web pages were called hypertexts and they were just static documents mostly containing text. Nowadays, the World Wide Web is what we commonly call the Web, and it is most frequently and incorrectly known as the Internet, the technological networking infrastructure globally connecting computers together in a massive network. A strong evolution changed the mechanisms and the appearance characterizing Web pages. Today, beyond text, they can contain images and videos and they became dynamic since new content can be quickly loaded using asynchronous calls.

The evolution of Web technologies strongly contributed to the creation and growth of an enormous number of services such as on-line stores and businesses, wikis, forums, search engines, social networks and so on. Among them, there are also the *Four Horsemen* of the digital economy[3], some of the most important companies in the World: Amazon, Apple, Facebook and Google. These companies are well known examples of producers of what has been called the "new gold of the 21st century", *data*. During the first decade of the 2000s, the amount of data generated and stored have become so massive that the term *Big Data* has been coined for identifying data sources that are so voluminous, various, quickly generated and varying in veracity that the traditional

[1]http://info.cern.ch/hypertext/WWW/TheProject.html
[2]http://timeline.web.cern.ch/timelines/the-birth-of-the-world-wide-web
[3]http://dld-conference.com/videos/XCvwCcEP74Q

methods for processing data result to be inadequate to handle them.

This thesis is mainly centered on large data sources and in particular it is focused on their visual representation. The visual representation of large amounts of data in an efficient way that could allow users to easily understand data and gain insights from them is in fact a well-known problem and a challenge in the research field of *Information Visualization* (InfoVis). This problem affects many kind of data producers such as companies and research centers that need their data to be analyzed, examined or investigated by humans.

While this study obviously can not address the huge variety of problems characterizing different disciplines and research areas, it focuses on one of those areas that can be identified as the *Semantic Web*. This term has been introduced by Tim Berners-Lee in 2001 as an extension of the traditional Web allowing machines to automatically process data by exploiting their new semantic and interlinked constitution. Later in 2006, he proposed *Linked Data*, a set of technologies for achieving the Semantic Web vision and allowing data owners to openly publish their data. The initiative had a good impact on the community, a nucleus of 28 data sets were published in 2007 and it further grew to 570 in 2014[4], today the LODStats[5] project counts 2740 active data sets overall containing 130.502.164.357 triples.

Linked Data sets are strongly impacted by visualization problems and represent the principal domain of application of this thesis. In fact, the design and development of new methods for visually presenting Linked Data sources in an effective way that allows humans to easily comprehend both their structure and content is widely investigated in the literature of this research area.

Since centuries ago, cartography has been dealing with large amounts of information, and today, computer-based solutions enhance this capability by interactively presenting even more complex and numerous data. It is quite common to be able to navigate geographic spaces and effectively perceive geographic information represented in cartographic forms. These humans cognitive skills can be also leveraged for exploring and analysing non-geographic data. Maps of abstract data resembling traditional geographic maps can be devised in order to activate those abilities. This *cartographic metaphor* leverages the visual perception abilities of humans and their consolidated map-reading skills for achieving a high level of efficacy in the communication of large amounts of complexly structured data.

The visualization approach presented in this thesis provides an entry point to the study of Linked Data sets and an interactive way for exploring them. It literally follows the *Visual Information-Seeking Mantra* of Ben Shneiderman [91], a very important infovis guideline in the design and development of visual representations. In order to produce visualizations leveraging on the cartographic metaphor, a *spatialization* approach has been embraced. This essential technique enables to “spatialize” abstract data by assigning them spatial coordinates necessary for displaying them on a two-dimensional space. The spatialization process has been carried out adopting a particular kind of fractal curve called *space-filling curve*, having the property of completely filling the available space. This specific feature has been used for assigning a position to the elements contained in the data set, and subsequently displacing them on the space forming

[4]`http://lod-cloud.net/`
[5]http://stats.lod2.eu/

a map. A novel technique exploiting the fractal structure of the curves has been devised for efficiently expressing data and enabling their spatialization in a very scalable way.

The resulting visualizations represent the abstract components constituting a Linked Data set depicted as if they were geographic elements collectively forming a map. The intended users are Semantic Web experts that can consult these visualizations for getting a gist of the generic characteristics of the structure of a data set (i.e., ontology and classes) but also precise information such as detailed features about resources and their relationships (i.e., instances and triples).

In the remainder of this work, Chapter 2 introduces the background of the two research areas involved in the thesis, that are Information Visualization and Semantic Web, and points out some not so conventional geometric notions. Chapter 3 presents the state of the art about the available tools for the visualization of Linked Data and the existing approaches for the spatialization of abstract data with particular focus on the ones based on space-filling curves. A preliminary analysis about some ordinary InfoVis techniques applied on Linked Data is given in Chapter 4. The main contribution of the thesis consisting in an approach for interactively visualizing Linked Data is presented in Chapter 5 while some specific case studies performed with existing Linked Data sets such as DBpedia and LinkedMDB are described in Chapter 6. Chapter 7 proposes some experimental studies on the concept of thematic maps, the possibility of placing elements according to their similarity and demonstrates the flexibility of the approach with an application prototype developed using the NCBI taxonomy data base. Finally, in Chapter 8, some conclusions about the whole study are drawn.

CHAPTER 2

Background

THIS chapter illustrates the background of the research area the thesis addresses: *Information Visualization* and *Semantic Web*. The former will be covered in Section 2.1 while the latter in Section 2.3. The main purpose of the chapter is to provide readers some essential notions, about the main topics that will be treated in the following, very important in order to better understand the details described in the core of the thesis. For this reason, also some not so ordinary concepts of geometry will be illustrated in Section 2.2.

2.1 Information Visualization

Information Visualization, also known as InfoVis, is a research field focused on the study of mechanisms for mapping abstract data in a visual form. The InfoVis outcome are the so called “visualizations", visual representations of data that amplify the human cognition. Properly designed visualizations help us in understanding data in an easier way, getting insights from them and revealing unexpected patters hidden within information.

The effectiveness of visualization in supporting humans is due to the strong human perception of processing visual information. In fact, it is well-known that the visual channel is the most powerful one in the human body. By analyzing the human visual cognition, InfoVis researchers can exploit this ability and produce proper visual representations.

It is important to specify some distinctions with other similar fields which information visualization is sometimes mixed. *Data Visualization*, also known as DataVis, is a broader concept, independent from computers. Indeed, several techniques for representing information, were already existing before the computer era [20, 21] at the end of the 19th century.

While InfoVis covers visualization of data that are often abstract in nature, *Scientific Visualization*, also called SciVis, deals with visualizations of physical data (e.g., geographical, architectural, meteorological, biological) coming from the real world [26].

Visual Analytics is a projection of InfoVis [109] and it mainly differs by strongly keeping the users in the loop of controlling the data mining, visual exploration and analysis process. Visual analytics tools are extremely interactive instrument enabling users to deeply analyse data through visual representations interactively created on their inputs.

The Information Visualization research field has become very important in the last few years in which data production has significantly grown. InfoVis techniques have been applied in different disciplines such as Social networks analysis, Machine Learning and Data Mining. Furthermore, several data analysis tools and libraries have been equipped with InfoVis methods for visualizing data. For instance *Tableau*[1], software for Business Intelligence and Analytics, is strongly based on data visualization techniques. *R*[2] [100], a software environment for performing statistical computing, includes several libraries for visualizing statistical data. The Machine Learning and Deep Learning library of Google, *TensorFlow*[3] [2] contains different kind of diagrams and charts for displaying the outcome of its algorithms. *Gephi*[4] [8] is a visualization and exploration platform for analyzing graphs and networks. *Tulip*[5] [5] is an information visualization framework dedicated to the analysis and visualization of relational data. Even some Web applications started to use InfoVis approaches, such as the *Newsmap*[6] service that is totally based on a treemap layout. Finally, *Data journalism*, a specialty of the traditional journalism very centered on data, is continuously growing. The most appropriate example is the *New York Times*[7] that often publishes interactive visualizations embedded within its on-line journalistic pieces.

2.1.1 Visualization pipeline

Generally, a set of data has to be firstly treated by several processing steps before being ready to be visualized. This stepwise procedure transforming data in a visual representation has been called *visualization pipeline* and described in literature by different works [26, 29]. Figure 2.1 shows a generic visualization pipeline composed of three main phases: *data transformation*, *visualization transformation* and *visual mapping transformation*.

The first step transforms raw data to a suitable abstraction. For example, a set of scientific articles can be transformed into a graph structure defining the citations between papers and the consequent relationships between authors.

The second step translates analytical abstractions into visual abstractions. It is a crucial procedure since abstract data are converted by enhancing them with information about their shape, position and possibly color. It is important to specify that this step does not produce any visual representation but it only equips data with the necessary

[1] `http://www.tableau.com/`
[2] `https://www.r-project.org/`
[3] `https://www.tensorflow.org/`
[4] `https://gephi.org/`
[5] `http://tulip.labri.fr`
[6] `http://newsmap.jp/`
[7] `http://www.nytimes.com/`

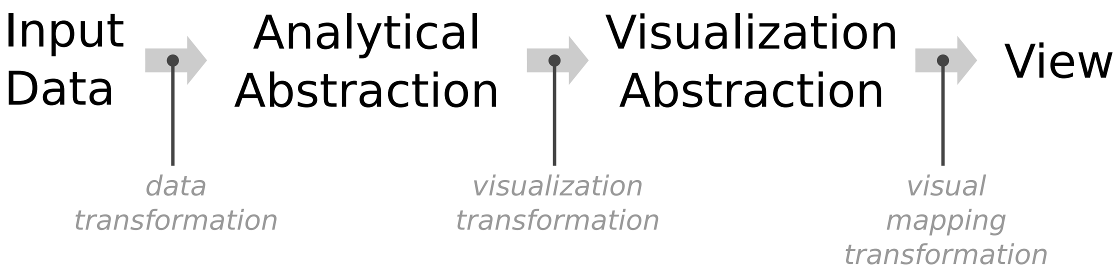

Input Data
Analytical Abstraction
Visualization Abstraction
View
data transformation
visualization transformation
visual mapping transformation

Figure 2.1: *A generic visualization pipeline showing the intermediate processes for transforming raw data into a visualization consultable by users.*

information needed for the subsequent actual visualization. The citation graph could be for instance visually represented as a *node-link diagram* where papers will be depicted as circles linked by lines defining their relationships. Hence, the position, radius and colour of the nodes and the curve the links draw are defined in this phase.

The last step produces a view displaying the final visualization filtered according to the user inputs. For instance, the overview of the whole citation graph could be demanded or only a small portion of it could be selected by the user.

2.1.2 Hierarchies

Hierarchies are often the base structure of many data sets and they have been widely studied within the literature of Information Visualization. In particular, by looking at the existing visual representations of hierarchical structures it is possible to notice that two basic graphical forms, which sometimes are also combined, have been mainly used: *stacked* and *nested* schemes [72]. Both forms graphically represent hierarchical relations between elements but in a different way: the former by using directional relationships (e.g., vertical, horizontal) while the latter by employing containment.

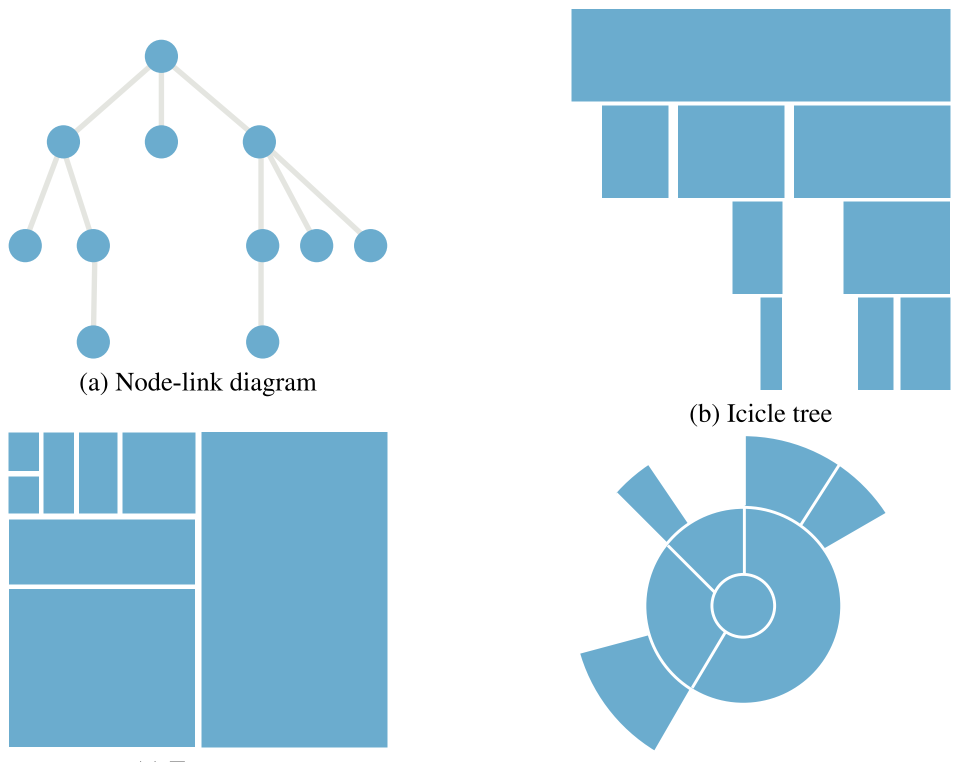
In this thesis, hierarchies play an important role as will be deeply illustrated in the next chapters. For this reason, it is worth to know how hierarchies are commonly represented in a visual way. A series of the most significant visualization techniques are briefly described in the following.

- *Node-link* (Figure 2.2 (a)) is the most commonly used layout for visualizing hierarchies. Nodes define the elements of the hierarchy while links represent the hierarchical relations between them. This technique has several layout variations, the most ordinary ones map the depth of the hierarchy on the y-axis and use the x-axis in order to separate siblings.
  - The *radial layout* places the root of a tree in the center of the representation and arranges the hierarchical levels around it at different distances. Even if hierarchical node-link diagrams provide effective overview of topologies, they result to be not so effective for visualizing large data sets due to their large aspect ratio;
  - Another variant, often used for representing the result of hierarchical clustering algorithms, is the *dendogram*;
  - Alternatively, the *indented layout* is widely used for representing hierarchical lists such as file systems.

- *Icicle tree* (Figure 2.2 (b)) [62] is a technique devised for improving the analysis of the results of hierarchical clustering algorithms. The elements of the hierarchy are drawn as solid areas (i.e., rectangular shapes) and their placement reveals their position in the hierarchy. In fact, given an element, it is possible to easily understand which is its parent and children by just looking at the adjacent rectangles close to it;

- *Treemap* (Figure 2.2 (c)) [53] is a method in which each element of a hierarchy is represented as a rectangle whose area is proportional to some attribute it owns (e.g., size, weight) [90]. The main idea behind this approach is to recursively partition the space into rectangular boxes as defined by the branching of the hierarchy. This approach is not as good as node-link diagram for analyzing topologies but it entirely fills the space available providing good results even with large hierarchies;

- *Sunburst* (Figure 2.2 (d)) [97] also known as radial icicle is the radial variation of the icicle tree. While the latter displaces rectangles along one direction, the sunburst places the elements of the hierarchy in a radial way in order to exploit all the degrees of the polar-coordinate system.



(a) Node-link diagram

(b) Icicle tree

(c) Treemap

(d) Sunburst

Figure 2.2: *Some well-known visualization techniques designed for representing hierarchical structures. On the top row, there is a node-link diagram and an icicle tree while the bottom row present a treemap and a sunburst.*

All these techniques are the most basic and commonly used. Upon them more sophisticated variations have been developed. A very wide and consistent survey about

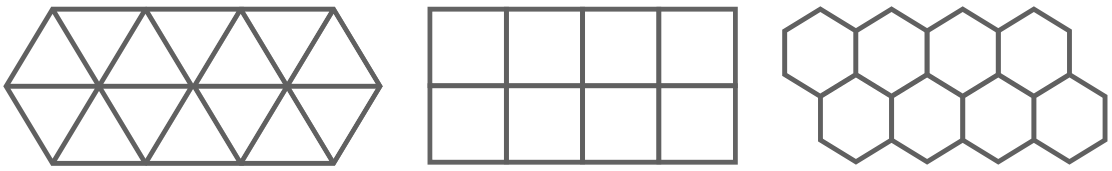


Figure 2.3: *The only existing regular tessellations are made by triangles, squares or hexagons.*

the visualization methods devised for representing hierarchies can be found on the *treevis.net* on-line bibliography[8] [87].

2.2 Geometry

In order to be ready for the explanation of the visualization approach proposed in Chapter 5, some not so conventional notions of geometry need to be illustrated.

2.2.1 Tessellations

A *tessellation* also known as a *tiling* is created by repeating one or more geometric shapes, called tiles, for covering a flat surface with no overlaps and no gaps.

Steven Schwartzman's *The Words of Mathematics* [88], says that the verb tessellate and noun tessellation derived from Latin tessera “a square table" that is in turn derived from the Greek tessares meaning “four" since a square tile has four sides. The diminutive of tessera was tessella, a small, square piece of stone or a cubical tile used in mosaics. Since a mosaic extends over a given area without leaving any region uncovered, the geometric meaning of the word tessellate is "to cover the plane with a pattern in such a way as to leave no region uncovered."

A *regular tessellation* is a tessellation constructed by congruent regular polygons[9]. There exist only three kind of regular tessellation made by triangles, squares and hexagons as shown in Figure 2.3.

2.2.2 Space-filling curves

This section gives more detailed information about a key concept of the approach described in Chapter 5: *space-filling curves* also known as *plane-filling curves* or *Peano curves* from the name of the Italian mathematician that in 1890 [79] discovered them. Initially, they were considered only as math curiosities. Then, in 1970 Mandelbrot coined the term “fractal" and published *The Fractal Geometry of Nature* [67], the foundation book that gave rise to a new mathematical discipline [106] and consequently to several studies on the topic. Space-filling curves are fractal curves having the property of passing through each point of a spatial plane. Examples of this geometrical shapes have been proposed by *Hilbert* in 1891, *Moore* in 1900, *Lebesgue* in 1904, *Sierpinski* in 1912, *Polya* in 1913 and others followed later [84]. Among them, it is important, for the aim of the thesis, to analyze the ones of Hilbert and Gosper since they will be consistently employed in the next chapters.

[8] `http://treevis.net`

[9] Regular means that the sides and angles of the polygon are equivalent while congruent means that the polygons have the same size and shape.

Let's start examining the Hilbert curve. A suitable way for understanding what a space-filling curve is exactly, consists in analyzing its construction process. First of all, it is necessary to consider its *generator* that is the most basic shape characterizing a curve. The Hilbert generator is the flipped-over U, shown in Figure 2.4 on the left-most side. The generator corresponds to the first *iteration* of the curve and to its starting point. The subsequent iterations can be obtained by recursively substituting the segments of the generator with scaled copies of itself. Thus, the second iteration of the Hilbert curve can be obtained by firstly substituting each segment of the first iteration with a scaled copy of the generator and secondly combining them in a joined curve. More precisely, the n-th iteration can be generally produced by recursively applying the following steps:

1. Given a square, divide it into four sub-squares;
2. Within each sub-square insert the Hilbert generator by properly selecting a reflection and rotation factor;
3. Connect the four generators composing the partial curve in order to obtain a continuous curve;
4. Perform the previous step on the four new sub-squares.

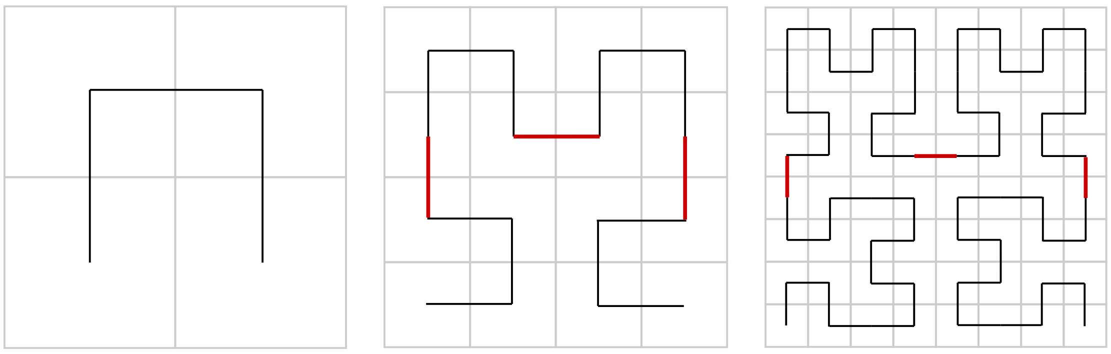


Figure 2.4: *The first three iterations of the Hilbert curve. The first one, on the left-most side, is also the curve generator. In fact, it is possible to see the flipped-over U repeated all over the curve of the subsequent iterations.*

The Gosper curve is a variant of the Koch Snowflake [60]. As the Hilbert one, it has the property of tiling the space but using hexagonal shapes. Differently from Hilbert, where a square can be easily split in four sub-squares, a hexagonal tile can not be perfectly divided into 7 new scaled tiles. In fact, by joining together 7 hexagons it is not possible to obtain a bigger regular hexagon. Only the approximation of a hexagon can be generated as a polygon tracing the exact shape of a regular hexagon.

Each iteration of the Gosper curve, has been called by Mandelbrot as *Gosper Island* [67]. Figure 2.5 shows the first three islands. The first one on the left is a single hexagonal tile, representing the smallest island. The first iteration in the middle is an island composed by 7 hexagons and the curve corresponds to the Gosper generator. The last one on the right is the second iteration comprising forty nine hexagons. By looking at the third iteration of the curve, shown at the right most side of Figure 2.5, it is possible to see that each of the 7 copies of the Gosper generator must accordingly placed with a certain degree of rotation in order to obtain the continuous curve joined by the red segments.

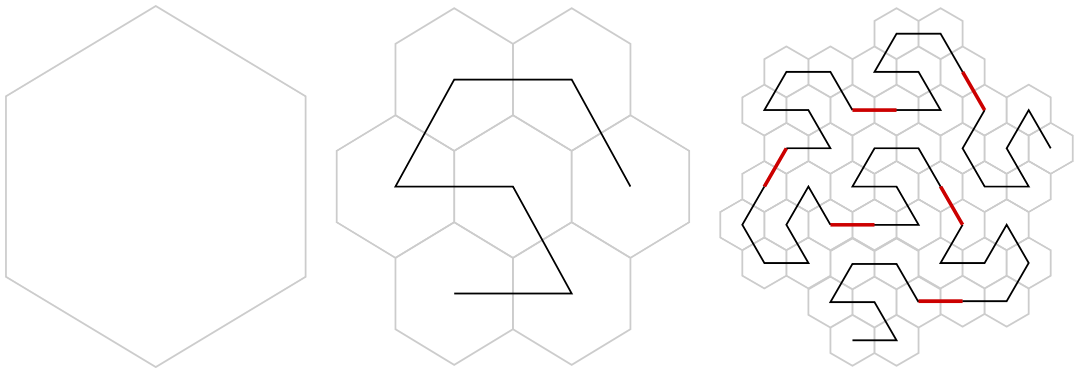


Figure 2.5: *The first three Gosper Islands. The first on the left is just a hexagonal tile while the two subsequent ones are respectively the result of the first and second iteration of the Gosper curve.*

Space-filling curves are a key components for our spatialization approach since they are adopted for the generation of regular tessellations. The fractal path the curves define

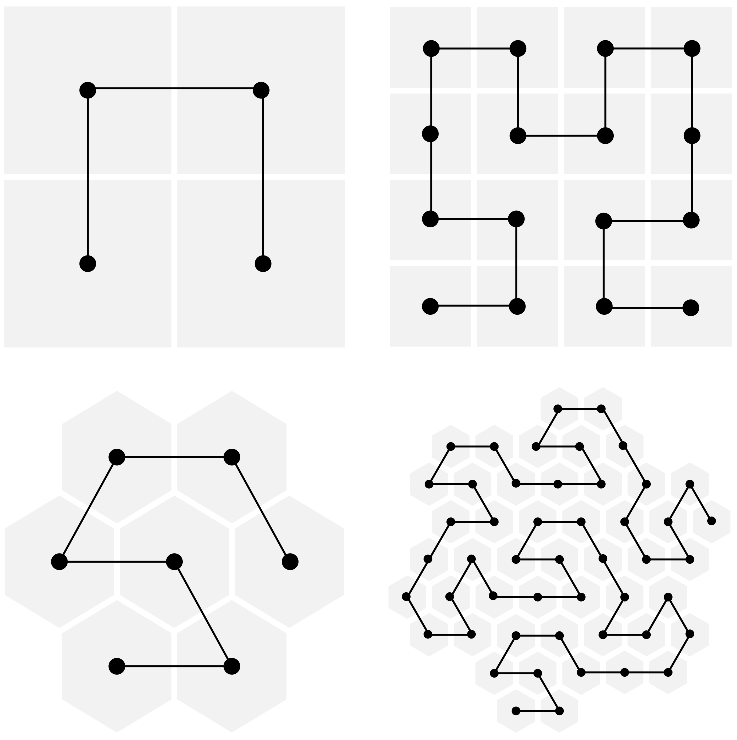


Figure 2.6: *The first two iterations of the Hilbert and Gosper curve showing how their path defines the coordinates on a flat surface in which tiles can be arranged.*

can be used for placing tiles on a flat surface. As shown in Figure 2.6, each vertex of a curve gives the exact point on which a tile can be placed. In both the Hilbert and Gosper cases, the result of the tile placing is a regular tessellation differing only for tile shape, square for the former and hexagonal for the latter.

So far the backgrounds addressing InfoVis and Geometry have been illustrated by introducing some key concepts such as visualization layouts and geometric notions. It is now fundamental to have some basic information about the Semantic Web and the technologies characterizing it.

2.3 Semantic Web

The term *Semantic Web* started to buzz the Computer Science research community in 2001 when Tim-Berners Lee published an article [13] in the Scientific American magazine. He presented a story about two siblings, Pete and Lucy, having to arrange some medical examinations for their mother by instructing some software agents exploiting the Semantic Web. The story was reported as an example, in order to present the strong distinction between the traditional Web of documents designed for humans and a new kind of Web in which machines could process the semantics of data: the Semantic Web.

In 2006, Tim-Berners Lee introduced the concept of Linked Data (LD), a set of technologies for pursuing the Semantic Web project which remained unrealized since its very beginning. Berners Lee presented a document [11] in which he illustrates some best practises for publishing data in a structured and interlinked way. First, he proposes to name things with URIs where things are any kind of entity that can be defined such as a person, an animal, a place, an object and so on. Second, he recommends to use HTTP URIs in order to provide useful information about things when their URIs are resolved by humans. Third, he suggests to link URI resources to other things so that data can be interlinked with each other. Finally, he chooses the employment of the Resource Description Framework (RDF) and the SPARQL language for respectively modelling and querying data.

The Linked Data project had a significant impact on the community and a considerable amount of data sets were published on the Web. Three years later, in 2009, Tim-Berners Lee together with Christian Bizer and Tom Heath summarize the progress about the initiative until that moment. 93 data sets, corresponding to 4.7 billion RDF triples, which were interlinked by around 142 million RDF links [16], resulted to be published. According to the data retrieved by the LOD cloud project[10], the number of published data sets reached 570 in 2014. The LODStats[11] project counts today 2740 active data sets.

Among the major data sets, DBpedia [7] is certainly one of the most interesting. The aim of the project is to convert the huge amount of unstructured information contained in Wikipedia[12] into a structured and semantic format. DBpedia extracts structured information from the infoboxes[13] within the Wikipedia pages belonging to different language editions. Nowadays, it is a reference data set for a large amount of other projects and studies within and outwith the Semantic Web research community. Other important data sets are Yago [99] that boast a large ontology derived from Wikipedia and Wordnet. Geonames [108] focuses on worldwide geographical information such as latitude, longitude, elevation, population, administrative subdivision and postal codes.

[10] `http://lod-cloud.net/`

[11] `http://stats.lod2.eu/`

[12]Wikipedia is a free online encyclopedia that, by default, allows its users to edit any article it publishes. It is the largest and most popular general reference work on the Web and is ranked among the ten most popular web sites.

[13] Wikipedia infoxes are fixed-format tables designed to be added to the top right-hand corner of Wikipedia articles to consistently present a summary of some unifying aspect that the articles share and sometimes to improve navigation to other interrelated articles (`https://en.wikipedia.org/wiki/Help:Infobox`).

Freebase [17] was a collaborative knowledge base of structured data released also as Linked Data in 2008. Then, in 2010 it was acquired by Google who used it to built its Knowledge Graph[14] and then closed it in 2014. Fortunately, Google decided to offer [80] the content of Freebase to the Wikidata community that integrated the data into Wikidata [107]. Wikidata is an on-line collaborative data source where user generated content, contrary to Wikipedia, is structurally inserted by users.

2.3.1 RDF

RDF is a framework for representing information on the Web [58]. It is built upon the *graph-based* data model that defines statements about web resources in the form of triples. A triple is a tuple composed by a *subject*, a *predicate* and an *object*.

- The subject could be an *URI reference* or a *blank node*. A blank node is a resource having no URI or literal, it has no name because it has no identity, it is blank. In some cases, it is useful to express statements in which a piece of information is missing;
- The predicate could be only a URI reference;
- The object could be an URI reference, a blank node or a *literal*. Literals are used to express values such as strings, numbers, dates and so on[15].

URI references identify individuals such as a person, an organization, an animal, an event, and so on. They are also known as instances or resources but their main purpose is to univocally identify an individual within the data set it belongs and also in the whole Semantic Web.

Sometimes the terms *datatype property* and *object property* are used for referring triples. These terms are used to describe what kind of value the predicate of a triple should have. More precisely, datatype properties relate individuals to literal data while object properties relate individuals to other individuals.

For instance, a triple having the predicate *hasName* would typically be a datatype property, since a name is a string, but *hasBrother* would be an object property, since a brother would be another individual.

In the following we list some basic examples of RDF triples in order to better clarify their components:

1. A triple is a tuple composed by a subject, a predicate and an object that can be written as follows:
   `<subject> <predicate> <object>`
2. The statement *"A guy named Gioele knows a girl named Ginevra and a guy named Carlo"* can be written as the RDF triples:
   ```
   <http://ex/Gioele> <http://ex/knows> <http://ex/Ginevra>
   <http://ex/Gioele> <http://ex/knows> <http://ex/Carlo>
   ```
   The subject, the predicate and the object are all URI references.

[14] `https://en.wikipedia.org/wiki/Knowledge_Graph` and `https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html`

[15] The complete list of datatypes available can be consulted at `https://www.w3.org/TR/rdf11-concepts/#section-Datatypes`

3. *"Gioele is 9 years old"*.
   `<http://ex/Gioele> <http://ex/age> 9`
   The object is a literal having an integer datatype.

4. *"Ginevra was born on the 21st April of 2012"*.
   `<http://ex/Ginevra> <http://ex/birth> "21/04/2012"`
   The object is a literal having a Time datatype.

5. *"Someone greets Carlo"*.
   `[] <http://ex/greets> <http://ex/Carlo>`
   The subject is a blank node that can be used for expressing the fact that someone, that we do not know and have no information about, greets Carlo.

6. *"Carlo eats something"*.
   `<http://ex/Carlo> <http://ex/eats> []`
   The object is a blank node for expressing that Carlo eats something but we do not know exactly what.

A collection of triples composes what is called an *RDF graph* and one basic example, composed of the triples listed above, is depicted in Figure 2.7.

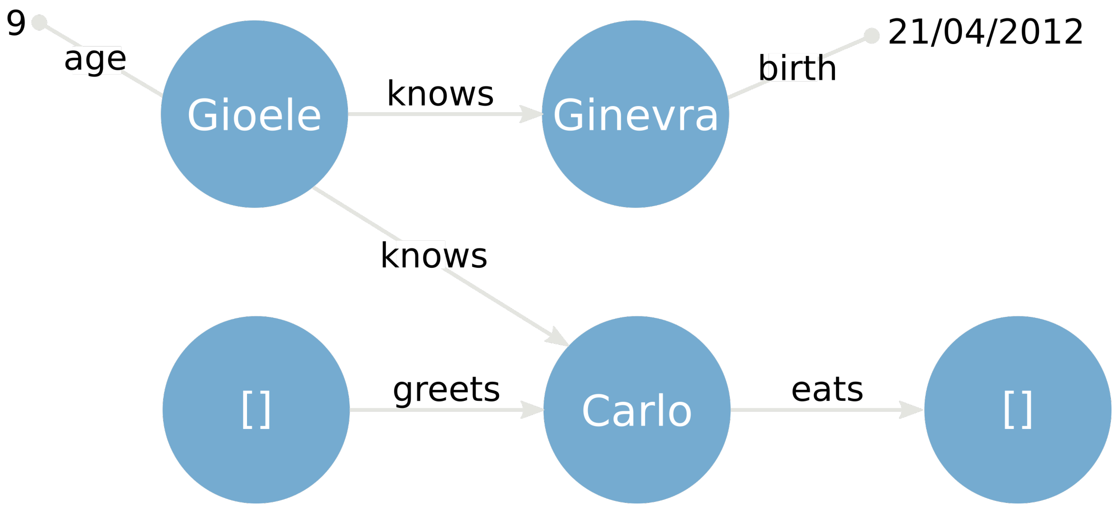


Figure 2.7: *The diagram shows the RDF graph composed by the RDF triples listed in the examples above. Blue nodes represent the subjects and objects that are URI references, small gray nodes are literal objects while gray edges describe predicates.*

2.3.2 Vocabularies

As described above, RDF provides a triple-based model for representing resources. However, it does not provide any specific terms for modelling resources but it only provides a generic framework. In order to specify which is the class of a resource or how two resources are connected to each other, we need to use *taxonomies*, *vocabularies* and *ontologies* [45]. Some of them are the Simple Knowledge Organization System (SKOS) [9], the RDF Vocabulary Description Language (RDFS) [19] and the Web Ontology Language (OWL) [33]. SKOS can be used for defining conceptual hierarchies while RDFS and OWL for describing the classes and properties of a set of resources.

Such technologies can be used in order to model new vocabularies. However, vocabulary reuse is a best practise. Hence, if a suitable term has been already defined in an existing vocabulary it should be reused instead of “reinventing the wheel" by defining a new equivalent term. Some of the most used and well-known vocabularies are:

- *The Dublin Core Metadata Initiative (DCMI)*[16]: it is a set of vocabulary terms that can be used to describe web resources such as documents, web pages, images as well as physical resources such as books, CDs and artworks;
- *Schema.org*[17]: it is a cross-domain vocabulary for structured data on the Web sponsored by Google, Microsoft, Yahoo and Yandex. It is widely used and employed by several applications from Google, Microsoft, Pinterest and Yandex;
- *The Friend-of-a-Friend (FOAF)*[18]: it is a schema for mainly describing people, organizations, and their own social network;
- *The DBpedia Ontology*[19]: is a cross-domain ontology manually created from the Wikipedia infoboxes and periodically mantained and updated[20];
- *The Bibliographic Ontology (BIBO)*[21]: it is an ontology providing concepts and properties for describing citations and bibliographic references such as quotes, books and articles.

Due to the large amount of vocabularies, we report only the widely known and used ones. A wide set of vocabularies and ontologies can be found on the on-line Linked Open Vocabularies service[22].

2.3.3 Storing and querying

In the above sections, we described how information can be represented using RDF (i.e., triples) and which are the instruments needed to model data and express semantics (i.e., vocabularies). However, so far we did not mention any process about storing and querying data. The key questions are now: “how can we store data?" and once stored, “how can we query them?". The brief answers are respectively *Triple stores* and *SPARQL*, two fundamental pieces among the Semantic Web technologies illustrated in the following.

Triple stores are databases specifically designed for storing and retrieving triples through query languages such as SPARQL [82], RQL [54] and TRIPLE [92]. There exist several implementations of triple store that can be classified by their features: *Native*, *RDBMS-backed* and *NoSQL* triple stores [89]. Some of the well-known triple stores are Virtuoso[23], AllegroGraph[24], Jena TDB[25], RDF4J[26], and Stardog[27]. Other

[16] http://dublincore.org/documents/dcmi-terms/
[17] https://schema.org/
[18] http://xmlns.com/foaf/spec/
[19] http://mappings.dbpedia.org/server/ontology/classes/ and http://wiki.dbpedia.org/services-resources/ontology
[20] http://bl.ocks.org/fabiovalse/0dfe7280086553c4a233
[21] http://bibliontology.com/
[22] http://lov.okfn.org/dataset/lov/
[23] http://virtuoso.openlinksw.com/
[24] http://franz.com/agraph/allegrograph/
[25] http://jena.apache.org/documentation/tdb/index.html
[26] http://rdf4j.org/
[27] http://stardog.com/

references can be found on the W3C[28].

Most of triple stores can be queried using SPARQL[29] as query language for interrogating RDF data. SPARQL has a similar syntax to SQL and it provides several operators such as OPTIONAL[30], NOT EXISTS and EXIST[31] and Property Paths[32] for performing complex queries.

[28] `https://www.w3.org/wiki/LargeTripleStores`
[29] `https://www.w3.org/TR/sparql11-query/`
[30] `https://www.w3.org/TR/sparql11-query/#optionals`
[31] `https://www.w3.org/TR/sparql11-query/#func-filter-exists`
[32] `https://www.w3.org/TR/sparql11-property-paths/`

CHAPTER 3

State of the art

DUE to the multi-field nature of the thesis, this chapter presents two different states of the art. The first one, covered in Section 3.1, is related to the existing applications and tools currently available for the visualization of Linked Data. The second one, covered in Section 3.2, is about the current research studies that make use of spatialization approaches with particular interest on the ones based on space-filling curves.

3.1 Linked Data visualization tools

Several surveys [30, 55, 68, 81] have been already published in order to evaluate which are the most efficient approaches and applications for visualizing Semantic Web data such as Linked Data or ontologies. The large amount of visualization studies developed within the Semantic Web research community can be explained by:

- the strong need for new tools for exploring these resources with effective and meaningful approaches. Typically, users approaching, for the first time, a Linked Data set find difficulties in understanding how the ontology is structured, which are the types of resources used and which are the most important ones;
- the peculiar characteristic of the visualization design process of answering to specific user's request with well-designed tasks. It is not trivial to create an approach that answers all the questions users are interested in. Sometimes it is more effective to construct distinct tools solving different problems.

The most complete and exhaustive survey by Dadzie et al. [30] presents a deep analysis of the to date available approaches for visualizing Linked Data, identifies their limitations, describes the visualization challenges which must be overcome and proposes some design guidelines and requirements which must be fulfilled in the design and development of such applications. In particular, together with the survey by Peña [81] and the Linked Open Data Visualization Model [22,23], it indicates the *Visual Information Seeking Mantra* of Ben Shneiderman [91] as a fundamental visual design framework for building information visualization applications. Shneiderman's Mantra states "Overview first, zoom and filter, then details on-demand" suggesting that the overview should always come first in a visualization, since it provides the general context of the data involved, and only at a later time users should be able to load more content and finally focus their attention on details. This guideline can be seen as the backbone of the approach presented in this thesis and described in detail in Chapter 5. For this reason, the Shneiderman Mantra has been used as a criterion for examining the works considered in the following analysis. The most significant techniques and applications, designed for solving the problem of exploring Linked Data, have been firstly analysed and then grouped in different classes according to their main features:

- *Graph-based* (Gb) comprises those works that are strongly depending on the *node-link diagram* visual representation;
- *Resource-oriented* (Ro) encompasses those works that are mainly based on the presentation of data related to single resources;
- *Query-based* (Qb) embraces those works that are mostly focused on the process of querying data set in a more intuitive way than the SPARQL querying language;
- *Other approaches* (Oa) characterized by particular features not classifiable with the previous categories.

3.1.1 Graph-based

In literature, a technique often employed for representing trees and networks, such as the ones characterizing the content and structure of Linked Data, is the *node-link diagram* where nodes are drawn as visual objects (e.g., they could be a shape such as a circle or a square, or an image or icon) and the links connecting them are drawn as directed or undirected lines [48]. Even if this visual idiom seems to be the most natural and well-suited representation for graph structures such as RDF graphs, it is not always the best one. In fact, as stated by Tamara Muzner in [74] node-links diagrams have the major weakness of scalability. They remain easily readable only for small graphs (i.e., dozens of nodes) and quickly degenerate into *hairballs* composed by hundreds of nodes. Moreover, they are suitable especially for those user tasks involving the network topology: find all possible paths from one node to another, find the shortest path between two nodes and finding all the adjacent nodes one hop away from a target node.

RelFinder [46] is a visualization tool specifically designed and developed for the task of finding relationships between two different resources in an RDF graph. It displays the result of a certain query, composed by the two target resources the user is interested in, as a node-link diagram where resources are nodes connected by links labeled and directed according to the RDF predicate they represent. Since the paths between the target resources could be composed by more then one hop with several intermediate nodes, interactive dragging allows to create a more readable layout with no overlapping nodes. This visualization tool results to be useful for finding which are the possible paths connecting two resources, however it presents the intrinsic scalability problem arising in node-link diagrams. Due to its specific purpose, the tool does not show the overview of an entire data set but only portions of it.

Aemoo [75] is another tool in which the employed graph visualization results to be effective for the specific task for which it has been designed. Analogously to RelFinder, it helps users in finding topologic information. In this case, the user can explore which are the adjacent resources directly connected to a certain target resource previously chosen. The target node is represented in the center of the diagram with a square shape. Its adjacent nodes, depicted as circles, are circularly placed around the target and connected to it through straight lines describing RDF predicates. In order to reduce the amount of nodes and produce a more readable layout, adjacent nodes are aggregated according to their class (e.g., rdf:type predicate). Even if this improvement streamlines the resulting graphs reducing the visual clutter, it increases the memory load as the user navigates beyond the target node. Moreover, the approaches hides to users which is the total amount of resources connected to the target.

Aemoo is not the only tool that simplifies the structure and decreases the size of graphs, in fact other applications such as **LODSight** [34, 35] try to reduce the intrinsic scalability problem of node-link diagrams. The tool is specifically designed for discovering data set issues through a manual exploration of their RDF graphs. Its approach consists in firstly summarizing the graph and then applying a node-link visualization technique. Moreover, the tool has a filtering feature that additionally limits the visualization only to those entities connected to the selected ontologies or properties. LodSight partially provides an overview of the data set since it includes classes, properties and data-types but it does not take into account instances.

Filtering techniques are often employed when dealing with large amounts of data, for instance **gFacet** [47] combines faceted filtering with graph-based visualization. This approach produces a node-link diagram where nodes are facets enclosing a set of resources that are aggregated in a group because they share the same conceptual structure. Besides connecting facet-nodes, links are themselves faceted since they allow to select a predicate among the ones defined for a specific facet-node. By selecting a predicate from a source node, its corresponding link becomes labeled and a new node is loaded as destination. This tool provides a mechanism for filtering resources and looking up their properties however it does provide a complete overview of the data set the user is exploring.

LODLive [25] is a graph-based application for exploring Linked Data sets. Given the URI of a resource, the tool starts the navigation of the data set the resource belongs. The initial resource is displayed as a circle node surrounded by tiny circles representing its object properties. By clicking on one of them, the tiny circle is exploded into a link connecting the initial resource and a new circle node representing another resource. The data properties of a resource can be loaded in an infobox by clicking on a particular button placed inside every circle node. This particular exploration approach is exactly the opposite of the Shneiderman Mantra since it results to be perfectly reversed. In fact, *Details on demand* are firstly loaded by asking users to specify the URI of a certain resource, *zoom and filter* are then performed by expanding object properties and *overview* can be obtained once all the resources are loaded into the diagram. The approach results to be effective for those user tasks that need a focused view about a certain resource. The main drawback of the tool is the difficulty of handling more than dozens of nodes once loaded by users.

RDFVis[1] [38] proposes an approach that computes in advance relevant information for enabling experts users to query the SPARQL endpoint of a certain data source. These information are all different classes of the resources of the data set, the kind of relationship connecting resources having different classes and which are the predicates used for modelling the resources of a certain class. In order to use the tool, it is necessary to pre-compute a JSON file containing the information of all classes present in a data set and the relationships between them. Then the tool can be activated and a node-link diagram visualization will be presented. Ontological classes are depicted as circles while the links connecting them as lines. Circles are depicted with a radius proportional to the number of instances the corresponding class contains. Links are displayed with varying width depending on the number of RDF triples defined with the predicate the represent. Ordinary nodes and links are coloured in blue while the ones having a subclass or representing more than one link are coloured in red. The tool provides a good overview of a data set in term of classes, instances and predicates. However, while it is possible to access to the information related to classes and predicates, instances are described only by their amounts.

Dadzie et al. [31] illustrates some challenges that must be addressed in presenting Linked Data. The most interesting one, regarding the exploration starting point, questions the appropriateness, of most RDF browsers, of require users start the exploration process from a specific URI. Our opinion, compliant with Shneiderman's, is that the focus of a specific resource, identified by a URI, should be provided only on demand at the end of the exploration process that should start with an overview. Moreover, users may not have the URI of the resource they are interested in. Hence, the overview supports users by providing them information about content such as the URI to start with. Other challenges concern the information overload arising when properties and relations of a resource are presented to user and the effective readability of complex data such as RDF. As a result of this analysis, the authors propose a template-based approach composed by different views like a node-link diagram.

The choice of representing RDF graphs with node-link diagrams motivated by the fact that they are graphs have been strongly criticized and described as "the pathetic fallacy of RDF" [86]. Moreover, it is well-known in the field of Information Visualization that this particular visualization idiom has the inherent and major drawback of not being scalable over more than few nodes. Hence, since RDF graphs typically range from thousands to millions of resources, we think that the best way for adopting this technique should be only for specific and properly defined tasks that give rise to a limited amount of nodes such as the ones addressed by RelFinder or Aemoo. Moreover, filtering techniques, nodes aggregation or a priori summarization could be additional methods for improving the resulting visualization and making it more readable and easy to understand.

[1] `http://jreutter.sitios.ing.uc.cl/VisualRDF.html`

3.1.2 Resource-oriented

Another characteristic that is recurrent in the applications presenting Linked Data is the focus on single resources. This kind of application is quite common within the on-line portals (e.g., DBpedia, Wikidata, GeoNames) where Linked Data are published and resources consulted by visiting their URI. These web applications mainly provide a view of a certain resource typically organized as a list in which RDF triples are reported. By consulting the triples of a resource, it is possible to navigate its data set by following the hypertext links of other resources.

This kind of application has been one of the first implemented after 2006 when Tim Berneers-Lee coined the term Linked Data in his Design Issue note [11]. Some examples are **Tabulator** [12], **Disco** [15], the **OpenLink data browser** [77] and **Marbles** [10].

More recent works have been performed in order to navigate resources and get data presented in a more user-friendly way. **LD viewer** [65, 66] is a framework for presenting the RDF data of the resources in a Linked Data set through an interface equipped by different features such as *pretty boxes*. These user-oriented and easy to use boxes display the most important properties of the viewed resource such as a picture, the title, the classes, a short description and the links to other resources. Furthermore, LD Viewer provides a search bar for looking up for resources, language filtering for choosing a preferred display language, triple filter for filtering triples using both properties and values, a *shortcut box*, live previews, maps for showing the geographic locations and *triple actions* for using external services (e.g., RelFinder, LodLive and the OpenLink Faceted Browser). **LodView** [24], provides similar functionality to the ones with which LD viewer is equipped.

In conclusion, even if these solutions results to be very useful for the task of consulting resource details, they have the significant drawback of being too focused on details without providing any kind of zoom and filter, and overview feature.

3.1.3 Query-based

Some applications dealing with Linked Data are strongly oriented to easing the process of querying data sets without the need of learning a Semantic Web query language such as SPARQL. This purpose is driven by the complexity of query languages and by the need of enabling non expert users in consulting information within Linked Data sets. **QueryVOWL** [42] proposes a visual querying system that defines a mapping between SPARQL and some graphical elements such as circles and lines already defined in the Visual Notation for OWL Ontologies (VOWL) [76]. The purpose of the tool is to provide a simple way, relying on recognition, for intuitively and effectively performing queries with visual objects and without handling complex SPARQL operators. By interacting with graphical elements, users can firstly add the resources they are interested in, and then bind them with relations. The resulting node-link diagram is automatically translated to a SPARQL query that is executed over a specific SPARQL endpoint. Even if the tool have the great advantage of letting users query a data set without knowing SPARQL, the graph-based approach limits the possibility of performing complex queries that would involve the use of particular SPARQL operators.

Other tools such as **Linked Data Query Wizard** [49, 50] prefer to present Linked

Data avoiding node-link diagrams using instead a tabular interface. Once a text search has been performed, a tabular view containing the resulting resources is provided. The properties of the resources can be loaded as new columns and filters can be applied in order to define the desired data set. Then, data can be visualized and consulted by choosing from statistical diagrams such as bar, pie and line chart, and more complex ones like parallel coordinates, stream graphs and mindmaps.

As QueryVOWL does, also **Gruff** [1] provides a graphical query view for creating SPARQL queries as diagrams of nodes and links. Moreover, Gruff allows to find arbitrary objects that are not easy to locate with popup menus for gradually honing the research.

VisiNav [43] is a resource-focused tool based on facets for constructing queries. Different operations are allowed: searching for resources of interest with a textual search, focusing on a resource and get its properties, getting information about other resources by following the URI specified by object properties, and manipulating result set by setting a facet (i.e., a combination of a property and a literal value or an object). Properties about resources can be visualized with lists, table views, timelines and maps.

Query-based solutions provide efficient ways for retrieving subsets of information contained in data sets and getting the details about resources. However, using these kind of tools makes not so effective to consult data overview.

3.1.4 Other approaches

The categories used in the analysis can not classify all the existing approaches the literature presents. In fact, other works specifically designed for solving particular tasks have been developed. **Spacetime** [102, 105] mainly provides a solution for visualizing data having both a spatial and temporal property. Once the user has selected a DBpedia class, a time range and a location the application returns the corresponding results displaying them both on a map and a timeline. The application provides also optional features such as time sliding animation and heatmaps. **Cubeviz** [36, 69] is a platform for visualizing statistical data. It provides different kind of diagrams, such as bar charts, line charts and radar charts, that can be configured by users in order to set the visualization according to their needs. **Payola** [57] is a web application for managing RDF graphs. It has a set of pre-installed plugins for visualizing data such as node-link diagrams, tabular views, treemap and maps. Moreover it allows to easily run analysis on SPARQL endpoints without the need of deeply knowing the SPARQL language. **Balloon synopsis** [85] is a node-centric RDF viewer available as a jQuery-plugin for the visualization of the RDF triples of the resources embedded within Web pages. By clicking on resources, an overlay appears on the Web page and RDF data are displayed using a tile layout containing also maps and simple charts. The **Dashboard-approach** proposed by Mazumdar et al. [71] consists of a panel in which several customizable visualizations, such as word cloud, map, pie chart, list, timeline, can be added in order to visualize data with different methods.

	Class	Overview	Zoom & Filter	Details on demand
RelFinder	Gb			x
Aemoo	Gb		x	x
LodSight	Gb	x		
gFacet	Gb		x	x
LODLive	Gb		x	x
RDFVis	Gb	x		
LD viewer	Ro			x
LodView	Ro			x
Tabulator	Ro			x
OpenLink Data Browser	Ro			x
Marbles	Ro			x
QueryVOWL	Qb		x	
LD Query Wizard	Qb		x	
Gruff	Qb		x	
VisiNav	Qb			x
Spacetime	Oa		x	x
Cubevix	Oa		x	x
Payola	Oa		x	
Balloon synopsis	Oa			x

Table 3.1: *This table summarizes to what extent each of the applications analyzed follows the Shneiderman Mantra. Each entry of the table is labeled with the class it belongs: Graph-based (Gb), Resource-oriented (Ro), Query-based (Qb) and Other approaches (Oa). None of them provide together the three characteristic of the guideline. Only one provide an overview. Some of them provide the zoom & filter feature. Most of them provide details on demand.*

3.2 Spatialization approaches

The *Spatialization* process is certainly the most important step in the pipeline of our visualization approach. Formally, it is the transformation of high-dimensional data into lower-dimensional, geometric representations on the basis of computational methods and spatial metaphors [95]. More generally, it can be defined as a process referring to the use of spatial metaphors to make sense of abstract concepts [96].

The *spatial metaphor* is the key-strength of spatialization approaches and in the recent years, it has been applied to represent non-geographic data for knowledge discovery within large and complex databases [96]. Spatial metaphors enable people that have previously learned how to read geographic maps to apply their own skills and gain new knowledge from abstract data having no inherently spatial attributes. This means, for instance, that databases about researchers, academic articles and citations could be visualized as maps. Other examples could be biological or medical data, stock market transactions or even Internet data flows. Possibly, every set of data, if accordingly prepared with pre-processing methods, could be visually represented as a cartographic map even if their attributes have no geographic meaning. In this section, it is illustrated that it is not required to have geographic data in order to properly place them on a two-dimensional space and in this manner create a map.

So, how is it possible to pre-process data in order to subsequently “spatialize" them? The literature proposes two main approaches for achieving this purpose: *dimensional-*

ity reduction and *spatial layout*. Among the major techniques following the former approach it is certainly important to mention the *Multi-dimensional scaling* [63] (MDS) and the *Self-organizing map* [61] (SOM). They both aim to create a mapping from high-dimensional data to low-dimensional data but while MDS needs only the distances between objects, SOM, that is an unsupervised and competitive neural network[2], requires as input their coordinates. On the contrary, the approach presented in this thesis is not based on dimensionality reduction but it adopts a treemap *spatial* layout approach, based on the use of a specific type of fractal curves called *space-filling curves*.

Space-filling curves are also known as plane-filling curves and they have the property of passing through each point of a spatial region. The concept was first introduced by Peano [79] when he discovered in 1890 the Peano curve. More detailed information about space-filling curves can be found in section 2.2.2.

Now, it is important to understand how space-filling curves can be used in order to provide a spatial displacement to the elements of a data set organized as a hierarchy. Basically, the fractal path a space-filling curve defines on a surface is used for arranging the set of elements. Each element is placed on a surface along the curve in a specific way that keeps its siblings in the hierarchy close to it. This process of spatialization, will be deeply described in Chapter 5. The result of this operation is a map-like visualization resembling a traditional geographic map that is instead composed by non-geographic abstract data. This particular feature has been called *cartographic metaphor* and consists in a powerful method for representing hierarchical abstract data in an intuitive way. In fact, the data are visually depicted as nested regions representing the hierarchical level of the hierarchy they compose. This metaphor eases the process of understanding information since it exploits the cognitive skills [37, 93] already developed by humans in reading traditional maps. The current approaches adopting the cartographic metaphors are briefly illustrated in the following, first the ones employing space-filling curves are presented.

In Auber et al. [6] a method called GosperMap for representing hierarchical structures is presented. The approach relies on the cartographic map metaphor and proposes an algorithm for producing treemap composed by irregularly shaped and nested regions. It appeals to the cognitive skills characterizing anyone who has ever learned to read a map. These skills are for instance the recognition of region containment and the pattern recognition of region areas. In order to construct a visualization supporting these skills, the method uses the Gosper space-filling curve for producing maps where regions have an area proportional to the sum of the area of their children. Beyond presenting the algorithm, the article proposes a method for separating region boundaries and better display nested regions so that hierarchies result to be more clear and visible. Furthermore, to fill the need of visualizing the evolution of hierarchies, it addresses the problem of the layout stability. Finally, an algorithm for placing labels within concave shapes is illustrated.

Vallet et al. [101] proposes JASPER, a particular layout for supporting users in the task of making sense of state changes occurring on nodes of large graphs. JASPER

[2]An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well. [98]

has been designed to compute a compact layout representing an overview of a graph. Differently from common graph layouts where both nodes and links are displayed, JASPER shows only the former hiding the latter. Moreover, it places nodes so that the adjacent ones (i.e. connected by an edge) remain as close as possible to each other, thus giving a hint of an existing connection between them. This solution firstly computes an initial and coarse layout of the overall graph by applying a clustering algorithm for reducing the number of nodes to be displayed and grouping them together. Then, the pixel-oriented[3] visualization is computed by ordering the nodes and placing them on the Morton space-filling curve (also known as Z-order curve). The main purpose of the resulting visualization is to assist users in keeping track propagation phenomena, basic situations arising especially in social networks similar to infections. Since the visualization is updated in real-time, these phenomenon can be visually tracked by viewers as color changes.

Wong et al. [110] presents a visualization technique, called GreenCurve, for interactively analyzing graphs and obtain their overview with maps. The first step for generating a GreenCurve is an ordering process that assign to each node a sequential number. Then graph nodes are folded into a space-filling curve. The Peano and the Hilbert curve have been chosen for their high degree of spatial coherence[4].

The work of Abrate [3] present a design concept called *Data Cartography* that mainly states that the cartographic discipline can be adapted to specific cases presenting abstract information. The idea is not only about positioning data on two-dimensional surfaces or keeping close semantically similar data. In fact, these aspects have been already presented in earlier works such as the ones illustrated above. The intent of the study is to communicate a large amount of complex information by embracing all the features characterizing Cartography such as visual representation methods for placing labels, drawing boundaries, using specific symbols, colors and textures. By combining these features with techniques such as geometric and semantic zoom, technologies such as topojson[5], geojson[6], and geospatial databases[7], the work presents a way for visually representing a data set in its entirety. Moreover, the study presents some applications of the approach using hierarchical structures and adopting a spatialization method based on different space-filling curves such as the Hilbert and the Gosper one.

Space-filling curves are not the only existing approach adopted for creating map-like visualizations. Biuk-Aghai et al. [14] introduces a method based on liquid-modelling. In order to build the final map-like layout, the method firstly produces a preliminary layout using the force-directed layout and overlap removal algorithms. The nodes position calculated in this step are then used as the input of the liquid modelling. The liquid modelling uses immiscible liquids in order to reach a stable state where nodes have their final position. The modelling starts with all liquids having equal parameters and stops when they reach a minimum threshold. Last step adds colours, borders and labels to the map.

In a previous work Biuk-Aghai and Yang [112] propose a method for visualizing hierarchical data in a geographic map-like form. After a preprocessing phase, for trans-

[3]Pixel-oriented visualization is a technique allowing to display large quantity of data in a minimal space. [56]
[4]The coherence level of a fractal curve is the amount in which neighboring pixels are at sequential positions on the curve.
[5] `https://en.wikipedia.org/wiki/GeoJSON#TopoJSON`
[6] `https://en.wikipedia.org/wiki/GeoJSON`
[7] `https://en.wikipedia.org/wiki/Spatial_database`

forming input data into a tree structure, has been computed the enhanced hexagonal-tiling algorithm starts. It is a recursive procedure that assign tiles for each node in the tree starting from the leaves till the root. Since the algorithm sometimes reaches situations in which no space is left for allocating an entity, a backtracking mechanism is used in order to solve the problem and find another position to accomodate the entity.

Skupin [94] presents a spatialization technique for creating map-like visualizations of knowledge domains. The main goal is to produces a multilevel visualization of non-geographical data in which both major and finer domains are displayed. The technique relies on cartographic principles and emulates traditional geographic depictions. The study present as example a geographic knowledge domain constructed analyzing the abstracts submitted to the annual meeting of the Association of American Geographers. The visualization methodology is mainly based on two well-known methods, *Self-organizing map*[8] (SOM) [61] for generating the two-dimensional coordinates of abstract data and Hierarchical clustering for grouping them in a nested structure enabling multilevel representations. The hierarchical clustering has been chosen as clustering method for the advantages it offers graphically, conceptually, and computationally.

Mashima et al. [70] improves the GMap method [39] with an algorithm variation handling dynamic data. It exploits the geographic-map metaphor for visualizing non-geographic relational data. Two use cases are presented, music trends of the Internet radio station last.fm and television viewing trends of an Internet Protocol television (IPTV) service. The GMap input is a graph of nodes and links. Firstly, the vertices are grouped using a modularity-based clustering algorithm. Then, the graph is embedded in a two-dimensional space using multi-dimensional scaling. The final map is then created using a Voronoi diagram of the vertices determined by the clustering and the embedding previously computed.

Gronemann et al. [41] proposes a method for drawing clustered graphs as topographic maps. The technique initially applies a slight variation of the Girvan and Newman clustering algorithm [40] based on edge betweenness. The algorithm calculates the betweenness for all edges in the graphs and then successively removes the one with the highest score until the graph becomes disconnected. The procedure is then repeated within each connected component in order to hierarchically clustering the graph. Finally the placement of the nodes is computed using the *fat polygon partitioning* [32]. It is a hierarchical partition scheme which differently from the typical partitions, such as the one used in treemap visualizations, uses convex polygons rather than just rectangles.

Another interesting study presented by Chalmers, sharing the intention of spatializing abstract data, has been carried out in the context of Information Retrieval. In fact, the traditional interaction techniques of this discipline offer access of information by means of isolated queries and word searches. Chalmers proposes Bead [27, 28], an approach displaying a document corpus in the form of a map or landscape constructed exploiting the similarity and dissimilarity of the documents composing the corpus. Bead represents documents as particles in a 3D space. By using physically-based modelling techniques the relationships between the documents are represented by their relative spatial positions. In fact, inter-particle forces tend to keep similar documents close to each other and dissimilar ones to move apart. The result of this physical process pro-

[8]Self-organizing map is an artificial neural network trained using unsupervised learning for performing dimensional reduction.

	Space-filling curve	**Tiling**	**Other approches**
Auber [6]	Gosper	Hex	/
Vallet [101]	Z-order	/	Pixel oriented layout
Wong [110]	Peano and Hilbert	Square	/
Abrate [3]	Gosper, Peano and Hilbert	Hex and Square	/
Biuk [14]	/	/	Force-directed layout and liquid modelling
Yang [112]	/	Hex	Backtracking
Skupin [94]	/	/	Self organizing map and Hierarchical clustering
Mashima [70]	/	/	Multi dimensional scaling, Modularity-based clustering, and GMap
Gronemann [41]	/	/	Edge betweenness clustering and Fat polygon partitioning

Table 3.2: *A summary of the spatialization approaches existing in the literature. Some of them adopt space-filling curves for the generation of the maps while others utilize other methods. The type of tiling used is squared or hexagonal.*

duces a 3D space relying on the metaphor of a landscape that allow users to interpret word-based information using the everyday phenomena of spatial position and colour.

CHAPTER 4

Preliminary study

THE preliminary study illustrated in this chapter, reports the initial analysis, performed on the existing visualization techniques, for devising the final approach described later in Chapter 5. The design process firstly examined the current approaches, available in the field of Information Visualization, for representing two specific kind of data set types: networks and trees. DBpedia has been selected as *test data set* in order to apply the different techniques found, analyze their properties and evaluate them. Using the same set of data over different kind of visualizations has been a useful approach for understanding which were the advantages and drawbacks of every technique.

In the following, we present all the steps progressively conducted in the analysis that enables the conception of our final approach. All the visualizations contained in the collection below have been implemented using the Data-Driven Documents (D3) Javascript library[1]. Furthermore, their code is openly published on to the Gist[2] service of the GitHub[3] repository. Due to the interactive essence of the visualizations proposed in the following, and for a better understanding of them, URL addresses for consulting their on-line version are made available within their relative image caption.

As it will be discussed later in more details, in most of the cases, the structure of Linked Data sets can be abstracted as a compound network that is mainly composed by a graph and a tree (more generically a forest). These two components are usually very unbalanced in term of size, in fact graphs typically result to be significantly bigger than their relative trees. For instance, the graph and the tree of the DBpedia compound net-

[1]http://d3js.org
[2]http://gist.github.com
[3]http://github.com

work have respectively 4 million and 7 hundreds thousands resources, and 721 classes[4]. For this reason along with the fact that trees describe the ontological and general organization of a data set, we started visualizing only this specific structural and hierarchical component.

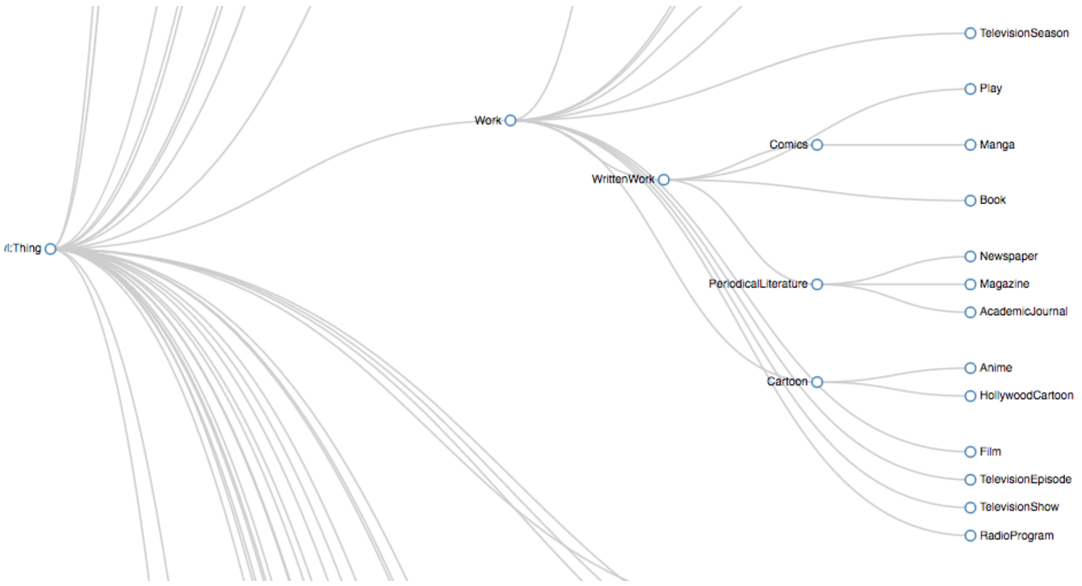


Figure 4.1: *A small portion of a dendogram representing the ontology of DBpedia. Nodes are classes represented as blue circles while the links denoting the ontological relationships between them are represented as gray curved edges. On the left, the outgoing links starting from the root node give a glimpse of the large amount of space this layout needs in order to be entirely depicted. The on-line version of the visualization can be consulted at* `http://bl.ocks.org/fabiovalse/c1f4cd4647dff0a412dd`.

The analysis starts with the use of the D3 cluster layout[5] allowing the generation of *dendograms*: node-link diagrams that place the leaf nodes of their tree at the same depth. The resulting representation, shown in Figure 4.1, displays ontological classes as labeled nodes and *rdfs:subClassOf* relations as gray curved edges. The diagram presents some restrictions: a scalability problem does not allow to entirely see the overview of the tree due to its large size. Furthermore, the layout produces overlapping edges that make worse their readability.

A simple improvement to the result given by the cluster layout can be achieved using the D3 tree layout[6]. This particular layout implements the Reingold–Tilford algorithm [83], a well-known algorithm used for creating efficient and tidy drawings, of aesthetically pleasing trees, with the minimum use of drawing space. Moreover, by introducing a collapsible node mechanism, implemented in the visualization shown in Figure 4.2, it is possible to obtain an interesting interactive result. The initial status of the diagram displays only the root of the tree and its direct children. The style of the circles denotes whether a certain node has children or not; a filled circle indicates the presence of a sub-tree that can be expanded by clicking on it. Even if collapsible interactions make the tree more compact, and the visualization more clear, this solution

[4]The numbers have been calculated from the 2014 dump
[5]https://github.com/mbostock/d3/wiki/Cluster-Layout
[6]https://github.com/mbostock/d3/wiki/Tree-Layout

contradicts the Shneiderman's Mantra since the overview of the data set is not anymore the first view presented to users.

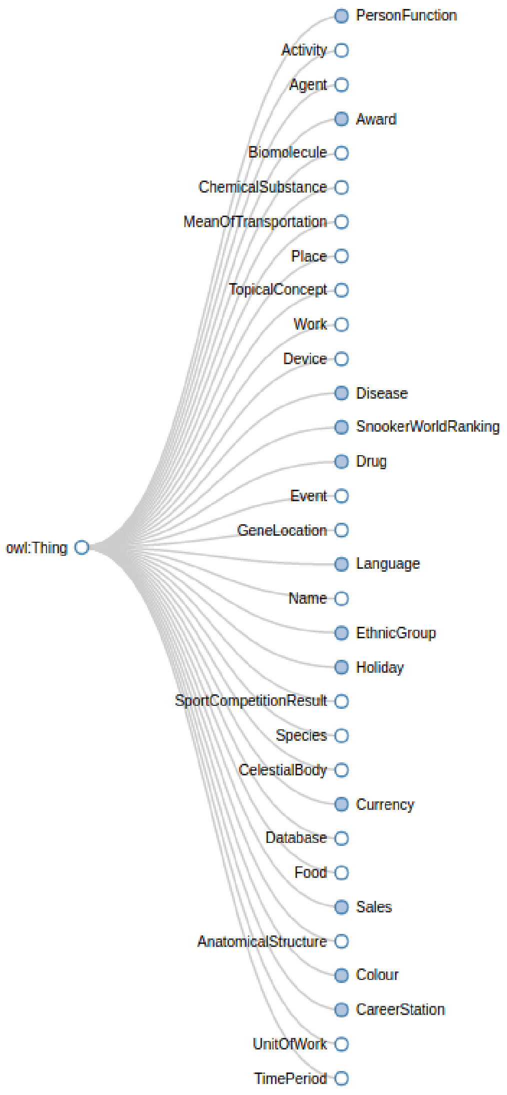


Figure 4.2: *The initial view of the collapsible tree representing the ontology of DBpedia. Nodes with stroke only and no fill represent intermediate nodes and denote the presence of a sub-tree that can be expanded on click while stroke nodes indicate the leaves of the tree. Clicking on intermediate nodes opens their relative sub-trees. The interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/d784198bdc1c76221393`.

Since it is important to be compliant with the Mantra, it is better to omit the collapsible tree just described and try to improve the dendogram illustrated above in Figure 4.1 even if it presents come scalability problems. Two new features are added: the Reingold–Tilford algorithm, already mentioned above, and a fixed ordering for the nodes of the tree called *canonical order*. According to this particular ordering, the branches of a tree are arranged by firstly preferring the ones with a higher depth. If two branches present the same depth, it is choosen first the one having more children. The outcome of this process, shown in Figure 4.3, is indeed characterized by a tidier layout. The canonical ordering strongly contribute in avoiding the links overlapping since it prevents that no edge is coincidentally drawn over the others. Generally, the visualization has a very clear layout but it still lacks on scalability since a very tall space is needed in order to fully contain it.

The obtained diagram can be further enhanced with additional information. For instance, as shown in Figure 4.4, the number of resources of every class can be encoded by the radius of the node circles.

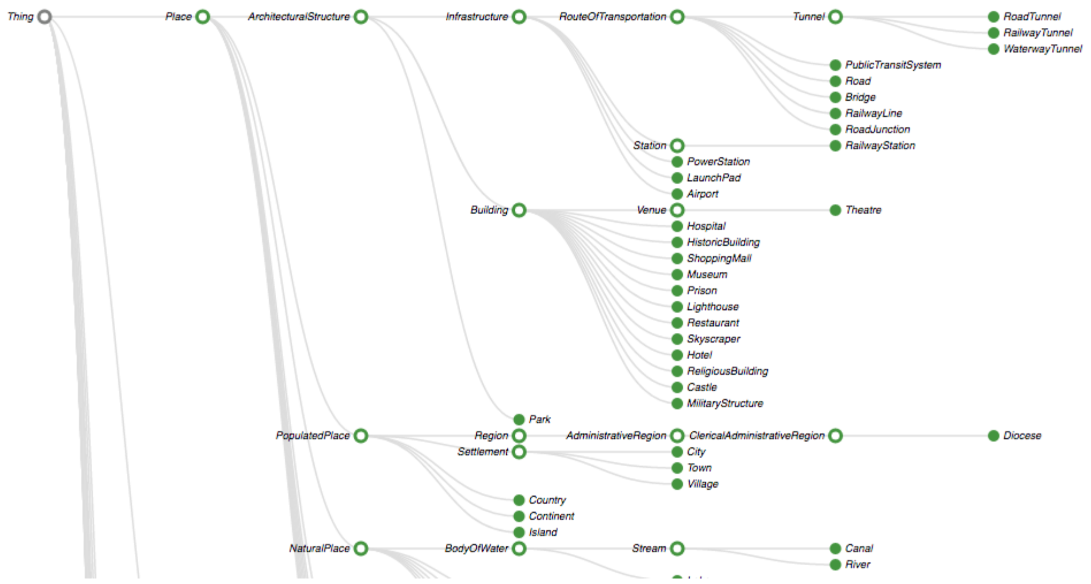


Figure 4.3: *A portion of the diagram showing the ontology of DBpedia. This particular visualization has been computed by applying the Reingold–Tilford algorithm and the canonical ordering. The layout is very readable since nodes are clearly arranged without any overlaps among the edges connecting them. The interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/25879cec40c9d8b08e6d`.

So far, only techniques for visualizing hierarchies has been presented. However, the most common kind of representation, used for visualizing networks and also trees, is the node-link diagram along with the force-directed placement. The force layout[7] implementation of D3 has been used in order to draw the graph and evaluate its properties. This algorithm assigns repulsive charge forces to the nodes for keeping them separated from each other. In addition, a pseudo-gravity force, keeps the nodes within the visible area of the visualization avoiding the expulsion of disconnected components. Links are not implemented with spring forces like in common force-layout algorithms, but geometric constraints are adopted for keeping fixed the distance between link endpoints. Figure 4.5 shows the application of this algorithm. Nodes represent ontological classes (e.g., Person, Organisation, Event, Places, Species) while links encode, in their width, the amount of connections (i.e., predicates of the RDF triples) between the instances of the class they are connecting. The result is quite scarce since the diagram is almost unreadable due to the disorder in the arrangement of the nodes and the links overlapping. This diagram truly demonstrates the scalability problem affecting this kind of diagram. Despite this restriction, several existing works employ this visual idiom for

[7]https://github.com/mbostock/d3/wiki/Force-Layout

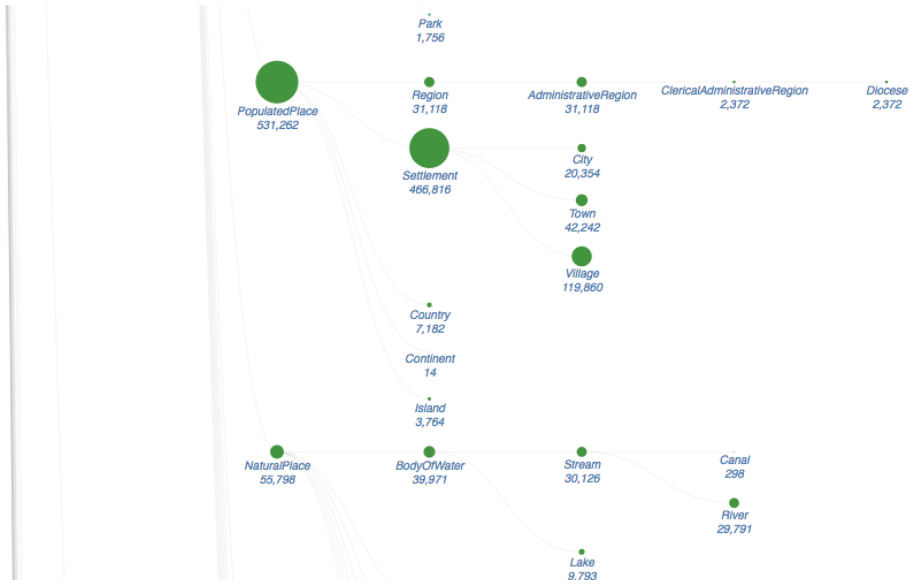


Figure 4.4: *A portion of the diagram showing the ontology of DBpedia. This particular visualization introduces an additional information to the previous one shown in Figure 4.3. In fact, the number of resources is encoded, for each class, in the length of the radius of the circle representing it. The web version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/1caa04f2c469c294e618`.

representing graphs such as Linked Data sets. Instead, other infovis studies state that is not suitable for such amount of data [74] and some Semantic Web works strongly disapprove it [86] by saying that the node-link diagram should not be used only because Linked Data are structured as a graph.

The inadequacy of the node-link diagram drives our analysis to continue the study of other hierarchy representations. Better nodes arrangement can be achieved using the radial layout in which the root of the tree is placed in the center of the diagram while each tree level is circularly disposed around it with a distance proportional to the its depth. As shown in Figure 4.6, the radial layout significantly improves the displacement of the nodes allowing to clearly understand the tree structure. This particular layout gets better results in term of aspect ratio since it drastically reduce the space needed for displaying the diagram in its entirety. However, it still remain quite disorganized due to the large amount of links that confusingly connect the nodes.

A well-known optimization for the representation of trees, is the *hierarchical edge bundling* [51], a technique, for reducing the visual clutter, that can be applied on top of standard tree visualization methods. Figure 4.7 shows the application of the bundling to the radial layout tree previously discussed. Edges are curved according to the hierarchical path starting from a node, rising to the root and going down to the destination node. The bundling consistently reduces the visual disorder of the links improving the layout. The result is quite interesting since it allows to get insights about the distribution of the edges within the hierarchy. In fact, by looking at the bundles formed by a significant overlapping of edges, it is possible to understand which are the most relevant flows of connections between the classes of the ontology. For instance, at a first look, it is very evident that there is an imbalance between the left and right part of the diagram.

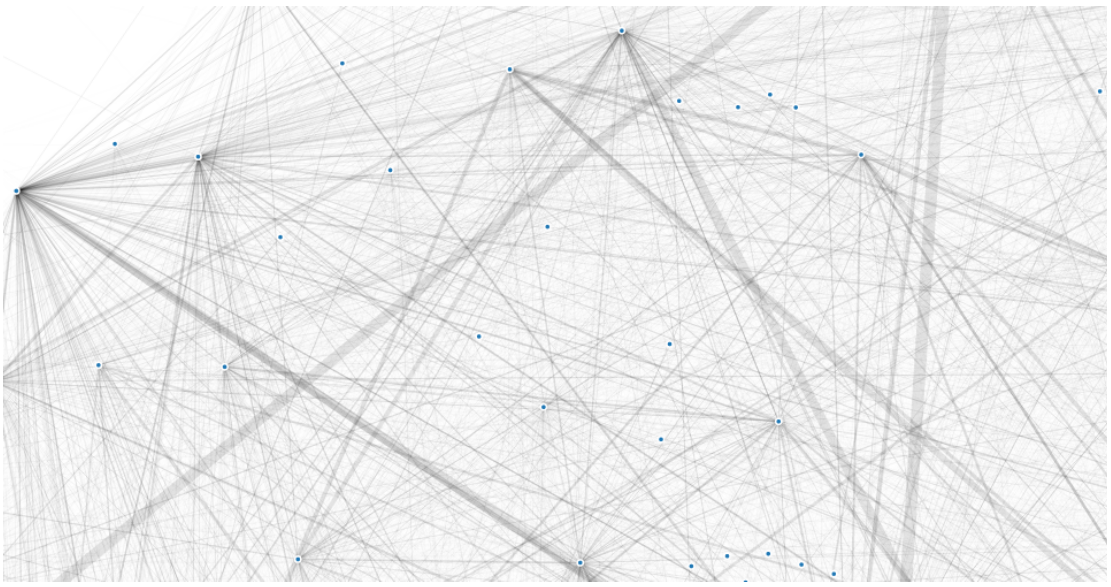


Figure 4.5: *A small part of the node-link diagram showing the ontology of DBpedia. The graph is very confusing since almost the whole space of the diagram is occupied by a large amount of links. The width of the links encode the number of connection between each pair of nodes. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/c2e578ce545305ebb95a`.

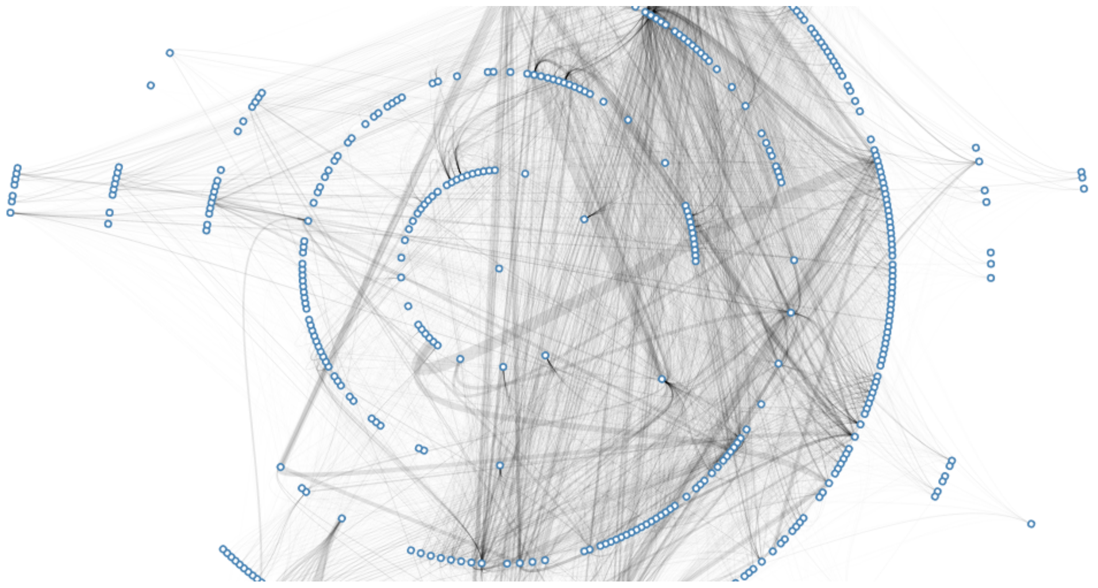


Figure 4.6: *The radial layout used for drawing the nodes of the ontology of DBpedia. The layout provides a more compact aspect ratio respect to the visualizations analyzed so far. The hierarchy can be seen more easily but links still remain very disordered and difficult to be read. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/02e0ee27f199416a8a0f`.

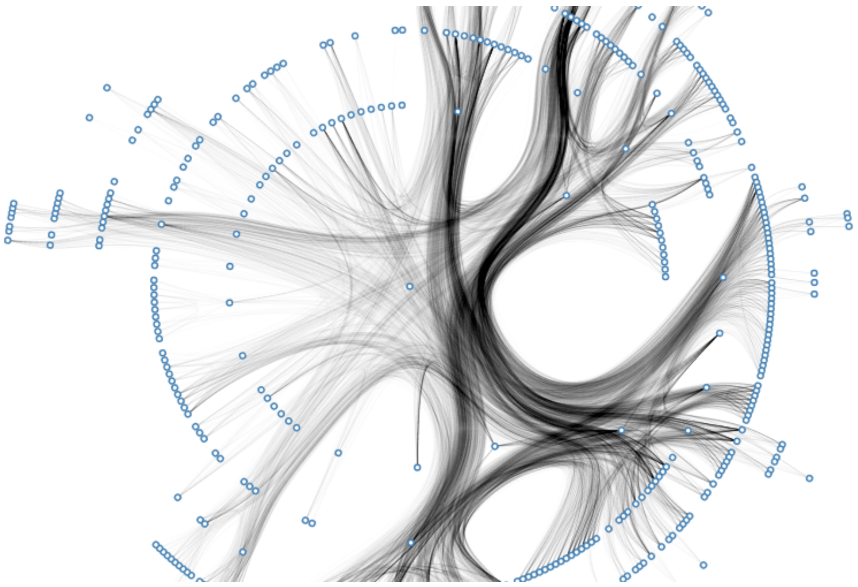


Figure 4.7: *The hierarchical edge bundling extremely improves the visual clutter characterizing the previous visualization. Furthermore, The bundles allow to get which are the most significant edge flows within the hierarchy. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/218d1d2f36125c64a32b`.

Moreover, the darker flows on the right side of the visualization mainly correspond to the edges between the class Person and the class Place or Work.

Alternatively, another technique that can be used for arranging edges in a visual pleasing way, is a novel method inspired by the Strahler number or Horton–Strahler number used for measuring the branching complexity of a tree. Basically, this technique groups links together forming the so called *pipes* that are aggregation of edges. Hence, all the edges passing through each pair of nodes are aggregated together forming a single edge encoding their amount with its width. From figure 4.8, showing the result produced by the technique, it is possible to see how the blue pipes summarize all the edges existing between the nodes. The visualization results to be too greedy since it does not consider the direction of the edges. The outcome, in Figure 4.9, has been obtained by distinguishing outgoing from incoming links that are separately aggregated and respectively represented with red and blue pipes. Moreover, the amount of instances belonging to each class is encoded on the radius of its corresponding circle.

A valuable variant for the representation of hierarchies with a radial layout is the *sunburst* technique also known as radial treemap. Differently from the previous visualizations, in which classes are drawn with circle nodes, this particular method represents them as radial arcs. The root of the hierarchy is placed in the center while the leaves are, the most external nodes of the diagram. Furthermore, edges are not represented at all, since the specific placement of the radial arcs reveals their location within the hierarchy. Then, the relationships between classes can be implicitly deducible from the arcs placement. In fact, a hierarchical relationship can be easily identified by the arcs connecting two nodes belonging respectively to two consecutive levels of the hierarchy. In addition, the amount of instances, each class owns, can be inherently encoded by the

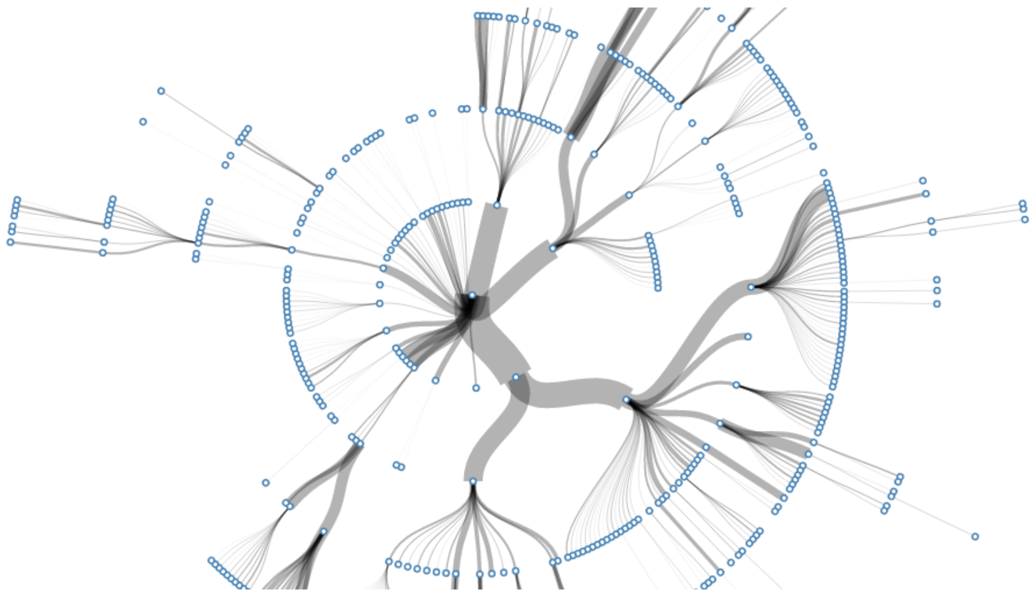


Figure 4.8: *A radial tree layout representing the ontology of DBpedia and adopting a method inspired by the Strahler number. Edges are aggregated together forming the blue pipes connecting the nodes of the hierarchy. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/1b3d58804fe02c62679d`.

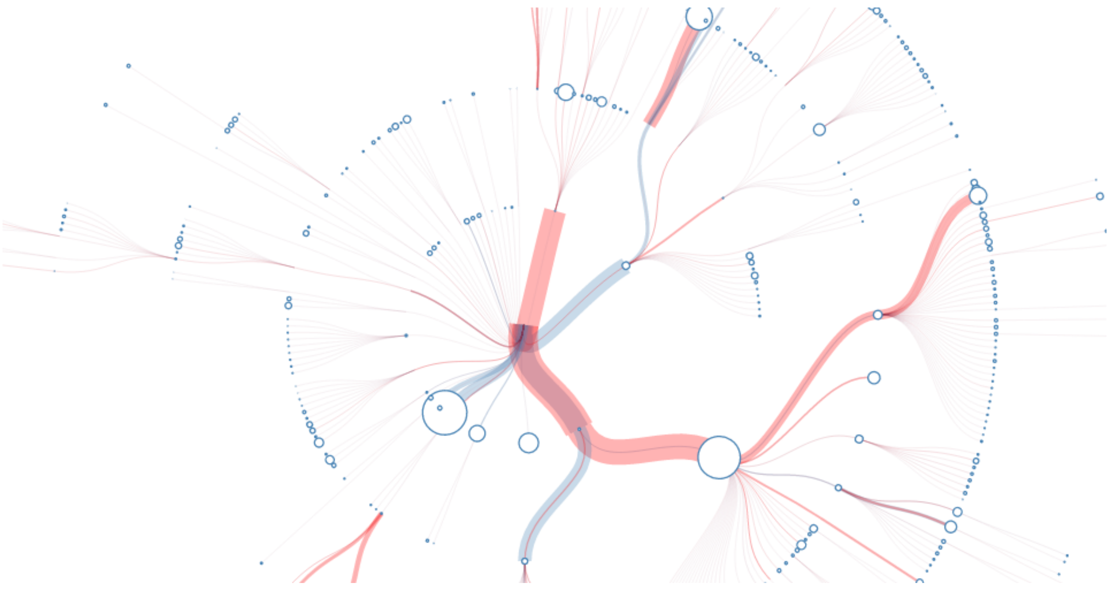


Figure 4.9: *Differently from the previous visualization, the direction of the edges are taken into account for distinguishing which are the outgoing and incoming links connecting the nodes. Moreover, for each class, its amount of instances is encoded on the radius of the circle representing it. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/672670526e0a9b642719`.

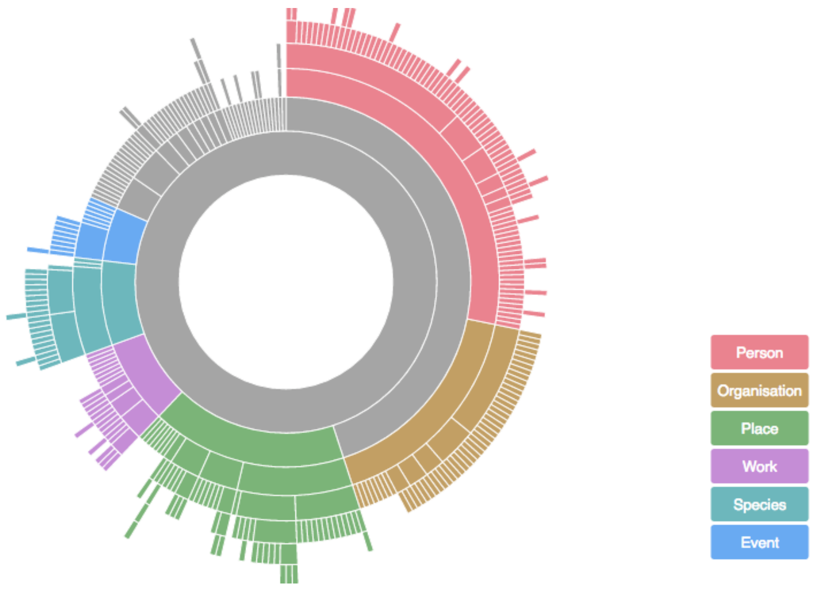


Figure 4.10: *The sunburst representing the ontology of DBpedia. This radial layout depicts nodes as radial arcs. The root of the tree is in the center of the chart while the leaves are the most outward arcs. Arcs encode the amount of children their sub-hierarchy contains. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/0dfe7280086553c4a233`.

degree of the angle the arcs embody. The D3 Partition Layout[8] has been used for producing the diagrams shown in Figure 4.10 and 4.11. While the former encodes, in the angle degree of the radial arc, the amount of children in the sub hierarchy of a certain class, the latter represents the amount of instances belonging to a specific class.

[8]https://github.com/mbostock/d3/wiki/Partition-Layout

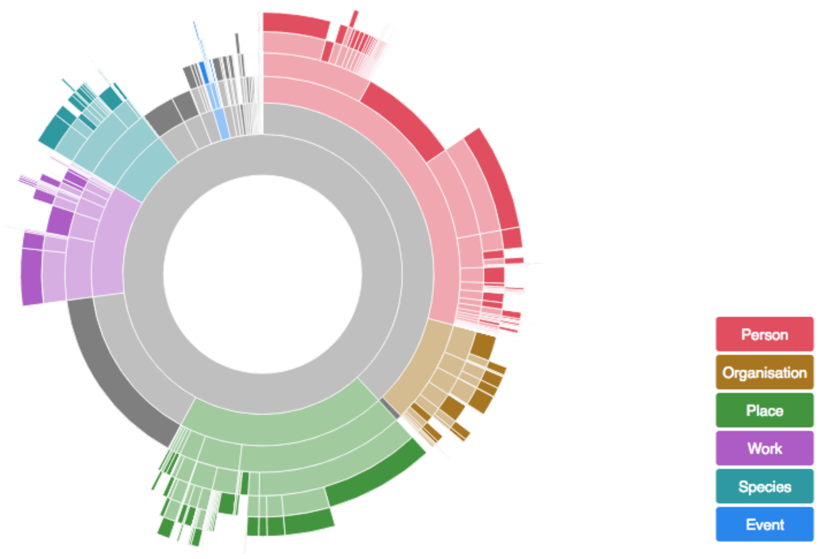


Figure 4.11: *Differently from the previous visualization, the arcs of the sunburst encode the amount of instances their corresponding ontological class owns. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/fa3be10877083f295204`.

CHAPTER 5

Visualizing Linked Data sets as abstract maps

THIS chapter illustrates the main contribution of the thesis, consisting in an interactive approach for the visual representation of Linked Data sets. As already shown in some previous works [103, 104], one of its main purpose is to provide an overall visualization representing entire data sets. The main idea is to initially provide a generic overview of the data set and secondly permit to explore precise portions of it for finally consulting its most specific details.

New visualization approaches are strongly needed for solving the comprehension problems that Semantic Web experts address every time they approach a Linked Data set. Especially for users having the need of performing queries, understanding how Linked Data sets are structured in term of ontology and how their instances are related to ontological classes are fundamental information.

In order to solve this problem, an approach founded on Cartography is presented. Since this discipline studies how to produce maps, depiction defining the relationships between objects in a space, we employ it for the visualization of abstract data instead of geographic ones. Even if data have no geographic meaning, they can be properly arranged on a two-dimensional surface in a specific way that resembles traditional geographic maps. This cartographic metaphor has been already adopted in the literature producing a high level of efficacy in communicating complex non-geographic data structure. The key strength of this technique is the reuse and exploitation of the map reading skills humans have already learned. In fact, the process of consulting a traditional map results to be equal to the one that can be performed on the resulting visualization our approach generates.

The use of a map as visualization enables to represent instances accordingly organized by the hierarchical structure defined by the classes of the ontology. In the resulting representations, ontological classes are depicted as regions generated by the aggre-

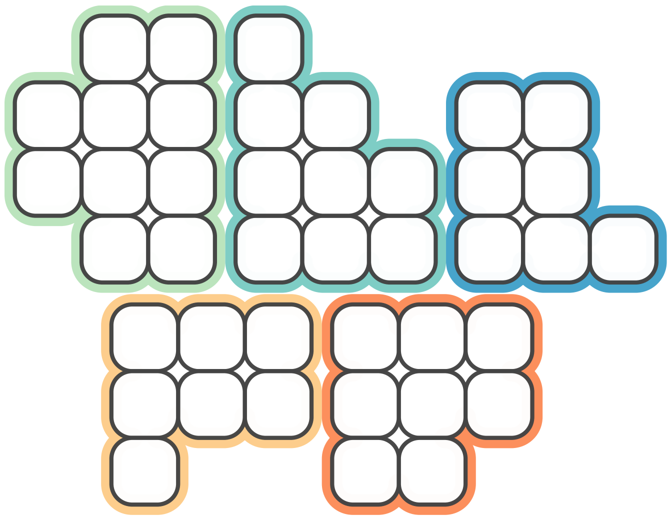


Figure 5.1: *A sketch sample of the idea behind our approach. Instances are represented as tiles while classes are given by sets of tiles.*

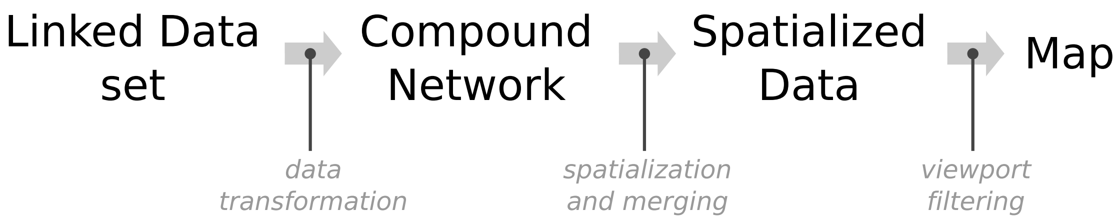


Figure 5.2: *The specific visualization pipeline especially devised for Linked Data sets that are gradually transformed into a map-like visualizations through three different steps: data transformation, sorting, spatialization and merging, and viewport filtering.*

gation of their corresponding instances. Thus, ontological classes such as Thing, Person or Animal are analogously represented as portions of space delimited by boundaries as in the case of ordinary administrative divisions like Country, Region or Province. As shown in Figure 5.1 Instances are the units of these particular maps since they are represented as tiles and placed on the space forming the regions corresponding to the classes.

In order to create such visualizations, a data set has to properly processed. For this reason, a specification of the visualization pipeline described in Chapter 2 has been instantiated for the case of Linked Data as shown in Figure 5.2.

A Linked Data set is the input data of the pipeline while the analytical abstraction of choice is the so-called *compound network* specifically described below. The visualization transformation step is composed by a spatialization and merging operation producing a set of spatialized data as visual abstraction. The spatialization process transforms abstract data to geometric objects by producing their preliminary shapes and positions. The subsequent merging procedure takes these geometric objects as input and merges them into bigger one, thus establishing their final configuration. The last transformation of visual mapping is controlled by a viewport filtering process enabling users to exclusively select the portion of data they are interested in.

In the rest of the chapter, all the steps for creating the visual representation, proposed by our approach, will be specifically illustrated by following the progressive structure of the pipeline.

5.1 Input data: Linked Data set

So far, the term Linked Data set has been only mentioned as the input of our approach without explicitly defining what exactly means. This section, present Linked Data sets in a more detailed manner by describing them in term of their structure, components and content.

As discussed in the Chapter 2, a Linked Data set can be seen as a graph data structure called RDF graph. It is simply a collection of RDF triples that is a list of elements in the form of subject, predicate and object. The reason behind its name, is that the triples compose a graph where individuals are the nodes and predicates define the relationships linking them. For example, considering the triples:

$$John\ hasMother\ Lucy$$

$$John\ hasFather\ George$$

$$John\ hasDog\ Rocky$$

$$John\ hasCar\ JohnCar$$

it is possible to obtain a graph whose nodes will be John, Lucy, George, Rocky and JohnCar connected by the links hasMother, hasFather, hasDog and hasCar.

An important distinction has to be made between the different kind of nodes composing the graph. Ontological classes are called *class nodes* while their instances, that are the individuals are called *instance nodes*. In particular, among the triples of a RDF graph, there are some that define the class of the individuals. The following example specifies the membership of the individuals to a class through a generic *isA* property. It is possible to see three classes that are Person, Dog and Car.

$$John\ isA\ Person$$

$$Lucy\ isA\ Person$$

$$George\ isA\ Person$$

$$Rocky\ isA\ Dog$$

$$JohnCar\ isA\ Car$$

Every Linked Data set determines the overall number of nodes it owns. A common aspect shared by these data sources is the strong imbalance between the class nodes and the instance nodes. Even some of the most important Linked Data sets are characterized by this lack of balance, as shown in Table 5.1.

Links differ for the kind of relationship they represent. In fact, it is possible to identify three typologies respectively defining a connection between two instances, two classes, and an instance and a class. For example, among the triples previously mentioned, hasMother and hasFather are relationships between instances while the isA predicate connects an instance to a class. Links connecting two classes could be the ones relating Dog to Animal, Person to LivingThing and Vehicle to NonLivingThing within a hypothetical ontology.

Dataset	Geonames	DBpedia	LinkedMDB	Wordnet	Lexvo
Class Nodes	7	721	51	5	5
Instance Nodes	8.3M	4.7M	694.400	647.215	128.945

Table 5.1: *The amount of classes and instances of some well-known Linked Data sets compatible with the data abstraction of our approach.*

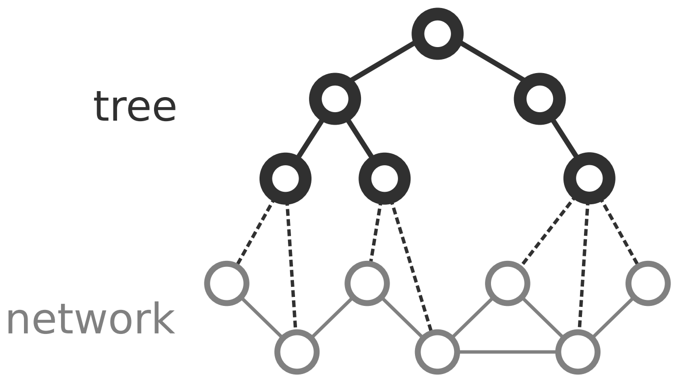


Figure 5.3: *An example of compound network constituted by a network (gray) and a tree (black). The leaves of the tree are the nodes of the network. Black links define the hierarchical structure of the tree. Gray links define the relationships between the nodes of the network.*

5.2 Analytical abstraction: compound network

The analytical abstraction is the result of the data transformation step. In our case, the Linked Data set input is converted into a *compound network* that can be generally defined as a network with an associated tree (Figure 5.3).

More precisely, the leaves of the tree represent the nodes of the network, while internal nodes provide a hierarchical structure for the leaf nodes that are connected to each other by the links of the network [74]. This kind of abstraction have been recently used also by other works [4, 52] in which approaches for the scalable drawing of networks has been illustrated.

Considering the purpose of our approach which visualizes instance nodes according to the structure described by class nodes, the abstraction defined by the compound network results to be a suitable solution. In fact, most of the time, RDF graphs are characterized by hierarchical ontologies organizing their classes and defining the base structure for the instances. More precisely, given the different nodes characterizing an RDF graph, it is well fitting with the abstraction to divide them in two different sets, instance and class nodes, where the former represents the graph and the latter the tree.

Some of the main Linked Data sets, results to be characterized by the compound network abstraction. For instance:

- DBpedia has a hierarchical cross-domain ontology whose classes organize a large variety of different instances derived from Wikipedia. Instances are linked together forming a graph according to the same connections Wikipedia presents.
- Geonames is a geographic data set containing instances representing place names. Its ontology is a hierarchy defining the divisions of administrative concepts such

as countries or regions.

- Several other cross-domain data sets can be abstracted as compound network if accordingly described by the emerging hierarchical ontology defined by schema.org.

Furthermore, not only Linked Data sets are organized in such a way. There are many cases that can be found in the Computer Science field, outside the Semantic Web research area, in which data sources present the same kind of organization, for instance:

- An academic publication network, structured as an institution hierarchically composed by different headquarters, departments, institutes and research groups, with the relationships between the publications of the researchers (i.e., citations of the articles);
- A software system hierarchically organized by directories, sub-directories and the files of the classes with the relationships defined by the dependency of a class from another;
- A social network composed by people belonging to different groups in different levels within a hierarchy, such as the organization structure of a company, with the relationships like the social interactions between people or the task allocation between colleagues;
- An outcome produced by the execution of Machine Learning algorithms such as hierarchical clustering in which observations are related to each other.

Hierarchical ontologies can be abstracted as a data structure widely used in Computer Science called *tree*. Generally speaking, a tree structure means a “branching" relationship between nodes, much like that found in the trees of nature [59]. More precisely, a tree can be recursively defined as a set of nodes, where one of them is the root and the remaining ones are disjointly grouped in sets that are in turn a tree and connected to the root. However,Linked Data sets are not always characterized by trees, in some cases, such as for LinkedMDB, Lexvo and Wordnet-RDF, a slightly different data structure, called *forest*, can be used as abstraction. There is a very little distinction between a forest and a tree. In fact, a forest is a set of zero or more disjoint trees or another way to phrase it is that a forest is the nodes of a tree excluding the root [59].

Let’s now consider a Linked Data set in term of a compound network. As depicted in Figure 5.4, two main components, a graph and a forest, respectively colored in red and blue, describe the compound network. There are two kinds of nodes:

- Class nodes compose the forest and represent the classes in the ontology;
- Instance nodes are the leaves of the trees and represent the resources (i.e., distinct URIs found as subjects or objects of RDF triples).

These nodes are connected by three kind of links:

- *V*ocabulary links are derived from the ontology of the data set and correspond to the *rdfs:subClassOf* predicate;
- *R*elationship links define the connections between instance nodes;

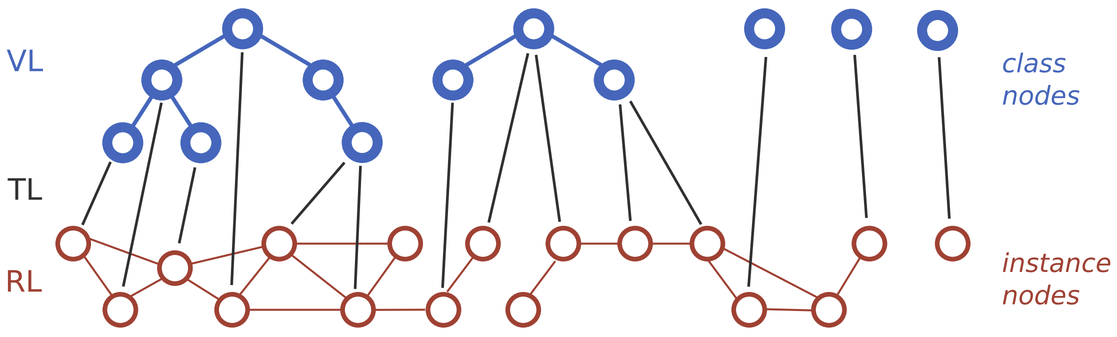


Figure 5.4: *A graph (red) and an associated forest (blue) define the extended compound network. In our case, it is composed by class nodes, instance nodes, vocabulary links (VLs), relationship links (RLs) and type links (TLs).*

- *T*ype links denote the membership of an instance to a certain class and correspond to *rdf:type* predicate.

In order to better clarify what exactly the compound network abstraction is, we give a very simple example shown in Figure 5.5. First, let's consider the forest. It represents the ontology of the data set. It is composed by two small trees, the first specifies persons with male and female genders, while the second defines places with two administrative divisions, regions and countries. The ontological entries in both trees are connected with each other through vocabulary links. For instance, the class *Male* and *Female* are linked to the class *Person* with the *rdfs:subClassOf* predicate. Let's now consider the resources or instances. Overall, they are five, two persons, *Fabio* and *Alessandra* that are respectively instances of the class *Male* and *Female*, three places, *Lombardy* and *Tuscany* instances of the class *Region* and one instance of *Country* that is *Italy*. By examining the relational links in the graph it is possible to understand which is the relationship between resources, for example, both Lombardy and Tuscany have Italy as country while both Fabio and Alessandra are connected to Tuscany since it is the where Fabio works and where Alessandra was born. It should be clear now which is the distinction between the graph and the forest. While the former represents the resources in an RDF data set and how they are connected, the latter classifies them with hierarchical structures defining the context of the data set. In fact, just by looking at the blue structure of Figure 5.5 it is possible to understand which are the type of resources the data set is composed, while by focusing our attention on the red part we can figure out which are the precise resources involved.

It is important to specify some details about the different kind of type links accepted by the abstraction. In general, an instance node can be a member of more than one class only if they are *compatible*. Two or more class nodes are compatible if they belong to the same branch of the hierarchy. For instance, by examining the previous example, the instance of Lombardy can be linked to Region and Place since they are is the same branch but it can not be connected to Region and Country since these classes are in two different branches. It is important not to violate this compatibility constraint because otherwise instance nodes would be connected to distinct conflicting class nodes, and consequently would have to be duplicated.

Another case regards instance nodes having no class nodes associated. This particular case has to be included among the compatible ones since input data could be

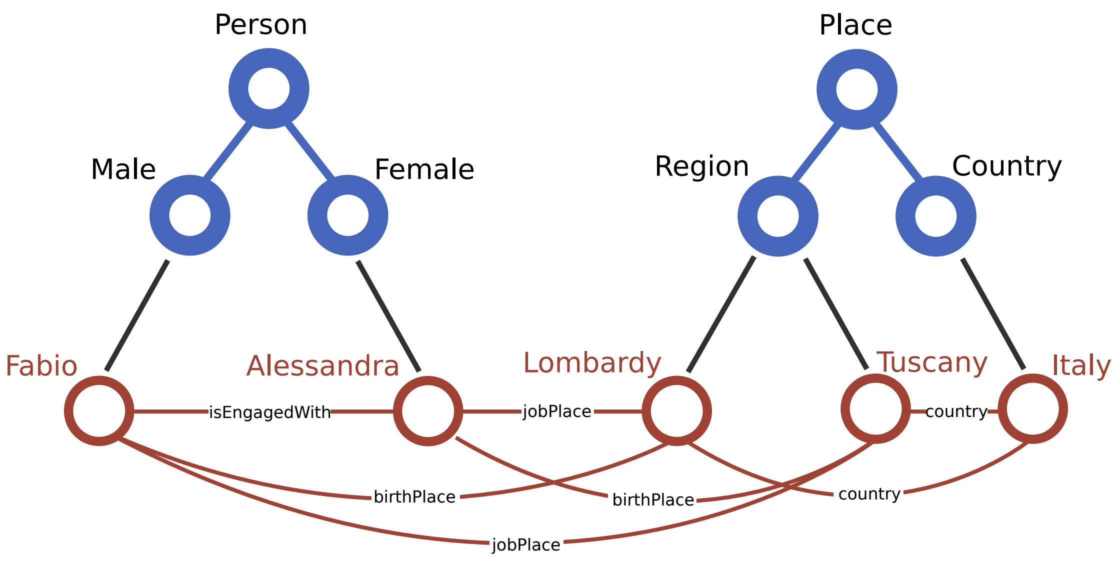


Figure 5.5: *An example of a small data set abstracted as a compound network. The forest (in blue) defines the structure with ontological classes such as Person, Place and their specifications. The graph describes their instances, that are the resources of the data set connected to each other with semantic relationships.*

inaccurate as in the case of DBpedia [78]. For example, the instance of DBpedia representing the Divine Comedy[1] in the 2014 dump is not described by any class of the ontology of DBpedia.

5.2.1 Sorting

Hierarchical data structures can be unordered or they can be generated according to a specific sorting criterion that sometimes can be not relevant and only an arbitrary feature. In the case of Linked Data ontologies, there is no inherent ordering that is worth to be preserved. In fact, considering an ontology in which the class Person and Animal are defined, there is no reason to take one of the two classes before or after the other. Even if there is no distinction between two copies of the same ontology ordered in different ways, their corresponding visualization could be very divergent. In fact, the sorting process is a simple but at the same time very crucial phase. Even if it is exclusively responsible for sorting the nodes of the forest defined in the compound network, it would cause serious stability problem in case it were not performed. In fact, it is an essential property, for a visual representation technique, to make their visualizations comparable among each other since it allow users to compare them and possibly gain significant insights.

For instance, suppose to have two distinct ordering of the same data source. If their corresponding forests were given as input to an order-preserving algorithm, substantial differences could be respectively generated in the outcomes even if no distinctions were present in the input except for the order. It is therefore fundamental to introduce a fixed ordering for preventing stability problems.

A possible way for ordering a hierarchy is using its topology features as done by the *canonical ordering* [111]. This sorting criterion provides a good stability and degree of

[1]`http://dbpedia.org/page/Divine_Comedy`

comparability. We employ it for handling *unordered forests* that are specific forests in which the order of the sibling nodes of every tree is not meaningful. This sort algorithm is solely based on topology features since its sorting criteria strongly depends on the depth of the tree branches.

More precisely, for each tree t, belonging to a given forest, the canonical ordering is applied. A recursive procedure $rsort$ consisting of a depth-first visit, is called by giving t as input. A comparison function $tcmp$ is used in order establish the priority between pairs of sub-trees belonging to t. Sub-trees are sorted according to the return value of the $tcmp$ function. Given two sub-trees a and b, $tcmp$ returns -1 if a comes first, +1 if b comes first, 0 if a and b are equal in term of their topology. The function recursively calls itself in order to find the first non-zero comparison result that is back propagated to the chain of recursive calls. In case two sub-trees are found to be equal, then the one having more children is placed first in the ordering by returning the difference between the sub-trees children amount.

```
rsort = (t) ->
  children = (if t.children? then t.children else [])

  for child in children
    rsort(child)

  t.children.sort(tcmp)


tcmp = (a,b) ->
  for (ai, bi) in zip(a.children, b.children)
    ci = tcmp(ai,bi)
    if ci isnt 0
      return ci

  return b.children.length-a.children.length
```

It is important to specify that the sorting algorithm discussed so far is applied only if no relevant order is applied to the input forest. For instance, the lexicographic order of the labels owned by the nodes could appropriate in some cases. Therefore, the sorting process is performed only when it is really necessary, otherwise the already established order of the input data is preserved.

5.3 Visualization transformation: spatialization and merging

The visualization transformation is responsible for the most crucial step in the visualization pipeline. The data abstraction of the compound network is transformed into a visual abstraction consisting in a set of spatialized data. The transformation mainly performs a spatialization process assigning a shape and spatial coordinates to both class and instances nodes.

5.3.1 Coordinates

In order to refer tiles on flat surfaces, cartesian coordinates are not the only option. Depending on the kind of tessellation and space-filling curve, a radix[2] can be selected for representing the numbers of a specific system of coordinates. For example given a squared tessellation generated by the Hilbert curve, a system whose coordinates are represented as base-4 numbers can be defined. Similarly, a hexagonal tessellation produced by the Gosper curve can be described by coordinates written as base-7 numbers.

Hilbert curve

A squared tessellation is required in the case of Hilbert. The relative coordinates system generates numbers with radix 4 that will be composed only by the composition of the 0, 1, 2 and 3 digits. For example, a valid number would be 021. The resulting coordinates are quite meaningful. In fact, their length reveals the number of iteration the curve has performed, since each digit is respectively assigned at every iteration.

By looking at Figure 5.6, it is possible to examine the coordinates of the first three iteration of the Hilbert curve. The first iteration is simply characterized by the coordinates 0, 1, 2 and 3 respectively identifying the four tiles composing the tessellation. In the second iteration, the number of coordinates increases due to longer path defined by the curve. Moreover, the generated coordinates, 00, 01, 02, 03, 10 and so on, have one more digit since the number of tiles to identify is increased too. It should be easier now to understand the increase of the coordinates length in accordance to the increase of the number of iterations.

It is interesting to look at how the tiles of a certain iteration are divided in the subsequent one. This specific division is the same characterizing a *quadtree*, a particular data structure often used for partitioning a two-dimensional space by recursively dividing it into four quads. Furthermore, by analyzing the composition of the resulting coordinates it is possible to notice that they acquire the digits of the quad from which they are generated with the addition of a new digit. For example, the coordinates 100, 101, 102, 103 of the third iteration start with the digits 10 inherited by the relative quad in the second iteration.

Gosper curve

In the case of the Gosper curve, a hexagonal tessellation is necessary to fill the space. The coordinates of the tiles are in the form of numbers expressed using the base 7. This radix is due to the specific division that has to be applied at each iteration of the curve. In this case, each hexagonal tile is divided into 7 scaled hexagons in the subsequent iteration.

Figure 5.7 shows the first two iteration of the Gosper curve. The coordinates generated in first one are simply the 7 unique digits (i.e., 0, 1, 2, 3, 4, 5 and 6) used to represent numbers in the septenary system.

Similarly to Hilbert, the coordinates of a certain iteration inherit the ones of the hexagon from which they have been generated in the previously iteration.

Differently from Hilbert where a square can be perfectly divided in 4 more little squares, 7 hexagons do not have the property of composing a bigger regular hexagon.

[2]A radix also known as base defines the number of unique digits used to represent numbers in a positional numeral system

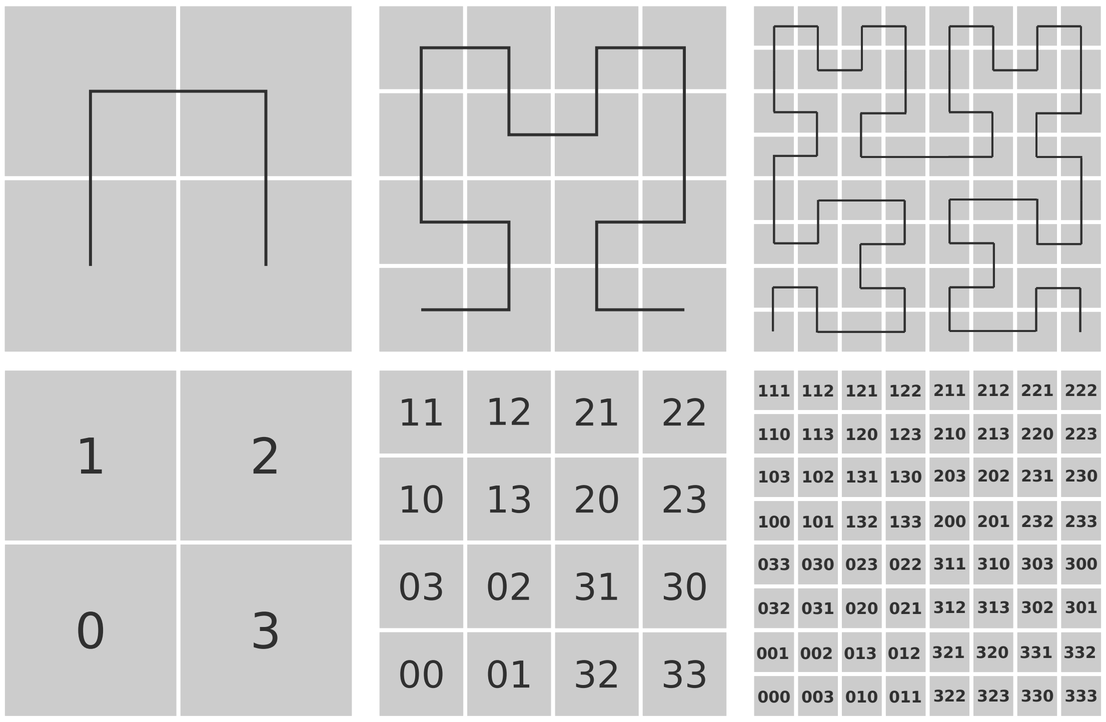


Figure 5.6: *The first three iterations of the Hilbert curve are shown from left to right. Each tile in every iteration can be identified by a number with radix 4. The length of the coordinates reveals the number of iterations performed. In the first iteration coordinates are 0, 1, 2 and 3. In the secondo iteration coordinates are 00, 01, 02, 03, 10 and so on.*

In fact, as shown in Figure 5.8 only an approximation of a hexagon can be obtained.

5.3.2 Spatialization

The spatialization process is the most important one in the visualization pipeline. Given the compound network, this phase proceeds to spatialize their instance and class nodes by assigning them a shape and a position.

For example, as shown on the top right side of Figure 5.9, each instance node is depicted as a squared tile placed at a certain position along the curve. Every tile is coloured according to the class its corresponding instance node belong to.

Due to the space-filling curve, the tiles of the same class are sequentially adjacent along the curve. In fact, it is enough to visit the tree of the compound network for determining a proper sequence of instances just by picking the leaves. This particular ordering allows to obtain a map in which tiles of the same colour are close together forming compact *regions*. The resulting map on the bottom right side is therefore composed by four regions representing class nodes. Furthermore, the *area encoding* technique can be observed within the map. Since the elements are represented with tiles of the same size, regions present an area proportional to the number of elements their corresponding class contains. Hence, by comparing the area of two or more regions we can figure out whether a region is bigger than the other in term of element amount.

So far, a general picture of how the spatialization process works has been presented. However, contrarily to what described above, our approach does not generate all the

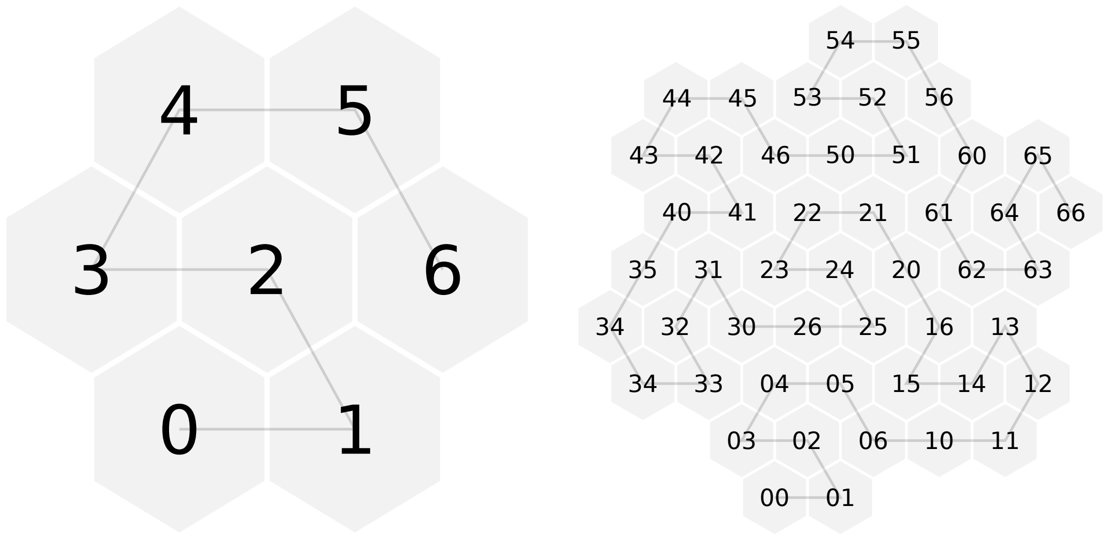


Figure 5.7: *The first two iterations of the Gosper curve represented with the tiles composing it and their coordinates. Numbers with radix 7 are used for identifying tiles.*

tiles before grouping them in regions. In fact, this particular procedure would results to be not so scalable with data sets having thousands or millions of instance nodes. For this reason, we devise a procedure that generates maps in a more efficient way without the need of generating large amount of tiles making costly effort.

Our approach is a novel technique exploiting the fractal nature of space-filling curves and their recursive composition. By taking advantage of the particular structure characterizing these maps it is possible to save a lot of computation for generating them. In fact, the resulting regions composing the maps can be partitioned into sets of shapes that exist in a finite number of configurations. These shapes called *macro tiles* are a scaled version of a tile produced as the aggregation of single tiles. As its counterparts, macro tiles have the property of tiling a surface with no overlaps or gaps but filling a larger amount of space. Thus, macro tiles compose a new set of tiles used for covering the surface in a more efficient and scalable way.

By looking at the simple example shown in Figure 5.9, it is possible to see that 64 tiles are needed for creating the map on the top right side and only 22 macro and single tiles (i.e., 10 macro tiles plus 12 tiles) for producing the map on the bottom left side. It is therefore very effective to generate macro tiles instead of single tiles, since the total amount of tiles (both single and macro) decreases. Moreover, the larger the data set is the higher is the difference between the amount of necessary macro tiles and the amount of single tiles that would be needed for filling the space.

The spatialization process can be described by a set of steps shown in Figure 5.10. Given the forest defined in the compound network as input, the spatialization firstly calculates the *order* of the map. Then, for each of the deepest classes of the forest, some parameters are successively computed by following a pipeline. First some *offsets* are calculated enabling the generation of *base-n numbers* that in turn allow to produce the spatial coordinates and size characterizing the macro tiles. The pipeline is illustrated step by step in the following.

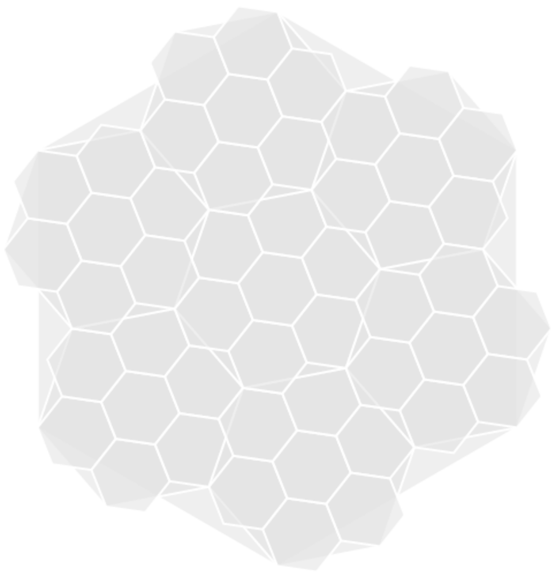


Figure 5.8: *The first three islands of the Gosper curve are overlapped one over the other in order to show how 7 hexagons do not form a bigger one but only an approximation. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/0670f38e8131b4f9e9da`.

Order

It is a factor defining the size of a map and the total amount of elements it does contain since it corresponds to the number of iterations a certain space-filling curve has been generated. Given the total amount of elements the input forest contains ($SIZE$) and a value identifying the space-filling curve and the type of tile employed (N), the $order$ can be calculated as the ceiling of the logarithm with base N of $SIZE$:

$$order = \lceil \log_N SIZE \rceil$$

In the case of Hilbert, N results to be equal to 4 since the curve fills the space with squared tiles forming a quad layout. In the case of Gosper, N is equal to 7 since the curve is characterized by a particular tessellation in which hexagonal macro tiles are recursively divided in 7 scaled hexagons at every iteration.

Given the order of a map, it is possible to calculate which is the maximum amount of elements that map can contain as N to the power of $order$:

$$max\_n\_elements = N^{order}$$

In the case of Hilbert, a map of order 1 contains at most $4^1 = 4$ elements, $4^2 = 16$ in a map of order 2, $4^3 = 64$ with order 3 and so on. In the case of Gosper, this value grow so much faster since N is equal to 7. An order-1 map comprises at most $7^1 = 7$, $7^2 = 49$ with order 2, $7^3 = 343$ with order 3.

We can say that the order defines an upper bound for the number of elements that can be contained by a map. For instance, for visualizing a data source composed by one million and five hundreds thousands elements, a map of $order = \lceil \log_4 1.500.000 \rceil = 11$ would be necessary in the case of Hilbert and of $order = \lceil \log_7 1.500.000 \rceil = 8$ with the Gosper curve.

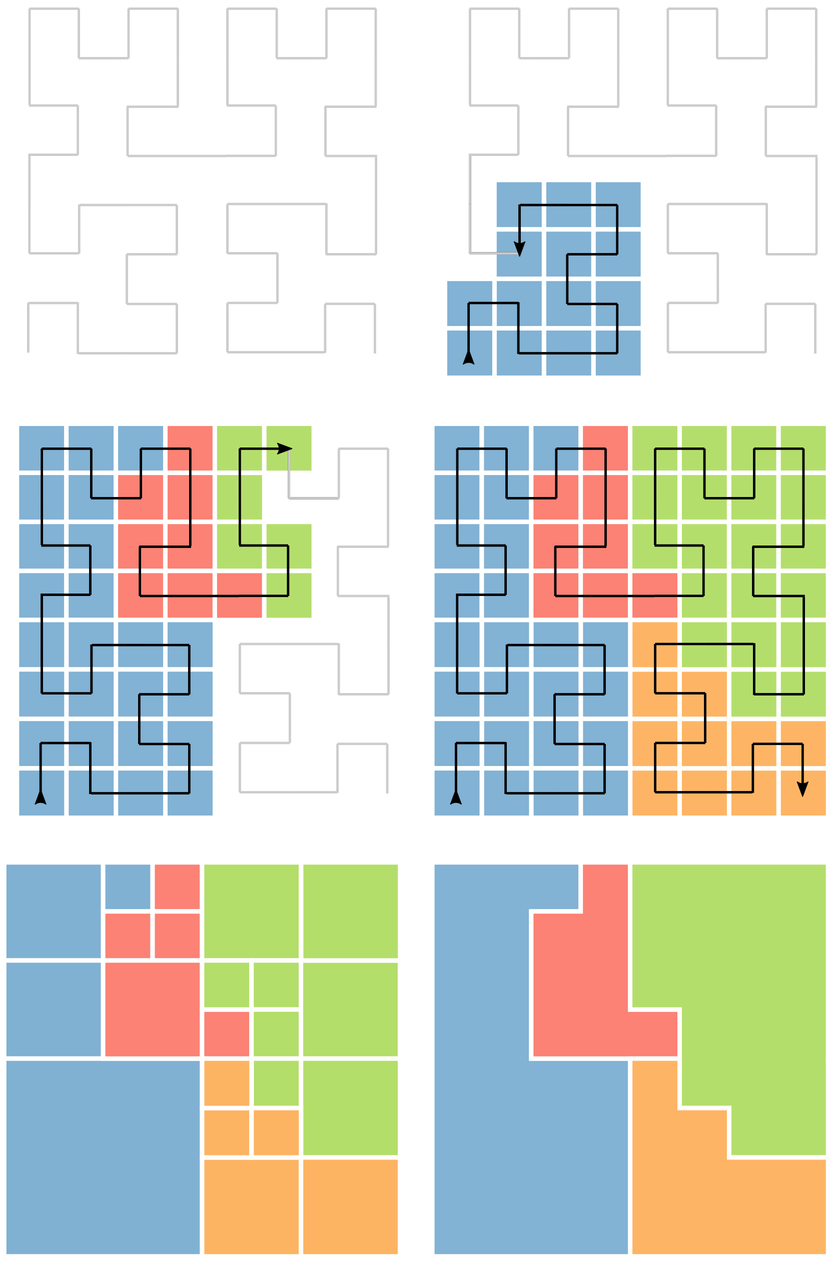


Figure 5.9: *An example of spatialization using the Hilbert curve and a squared tessellation. The image contains 6 sub-images describing the process of spatialization. The top left sub-image is the third iteration of the Hilbert curve. The first four sub-images show how the curve has been used for displacing instance nodes on the surface. Instance nodes are coloured according to the class they belong to. On the bottom left side macro tiles are shown instead of singles tiles. On the bottom right side the resulting map is depicted as the composition of its regions.*

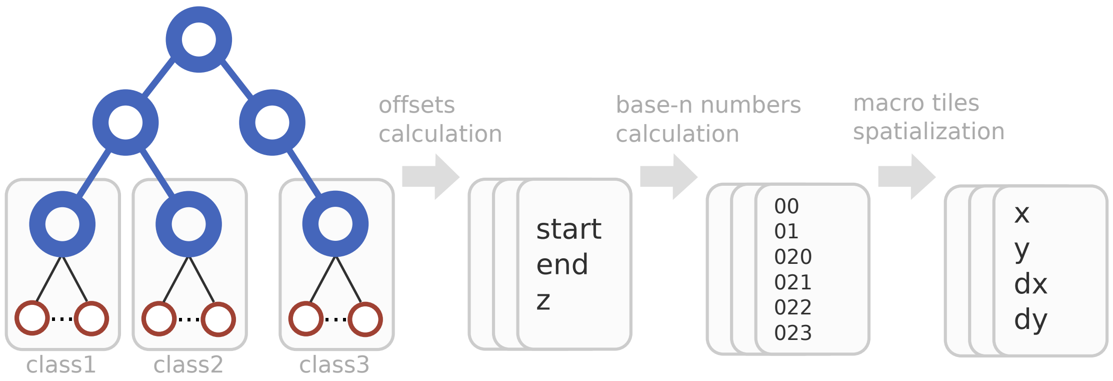


Figure 5.10: *For each of the deepest classes of the forest, three offsets are calculated (i.e., start, end and z). Then, the coordinates of the macro tiles, in the form of base-n numbers, are computed. Finally, the macro tiles are produced by generating a spatial coordinate and a size.*

Offsets calculation

In order to arrange the instance nodes on a surface along a certain space-filling curve, it is necessary to define a linear sequence that keeps the elements of the same class sequentially close to each other. For each class defined in the input, three offsets, calculated from the beginning of the sequence, have to be calculated: *start* (s), *end* (e) and z. s and e respectively denote where a class starts and ends within the sequence. A simple loop iterating over the classes can be used for calculating them. For each loop iteration, the amount of elements of a class is incrementally summed.

```
classes.forEach (c) ->
  c.start = start
  c.end = start + c.values.length
  start += c.values.length

  c.z = get_z c
```

The z offset requires a more complex procedure. In order to explain it, it is helpful to analyze the structure and the composition of the resulting regions in term of macro tiles. As previously discussed, the spatialization process produces regions as sets of macro tiles that follow the path of a certain space-filling curve. By unwrapping the macro tiles placed on the curve and by placing them on a straight line, as shown in Figure 5.11, it is much simpler to understand how a region is actually composed. Except for the first and the last one, regions are constituted by a sequence of macro tiles with size that is small at the beginning, increases in the middle of the sequence and decreases at the end. Hence, each region can be seen as the composition of two *“tails"*. The first starting with small macro tiles and finishing immediately before the first biggest macro tile. The second tail starting with the first biggest macro tile and finishing with the last macro tile of a region.

For instance, the red region is composed by eight macro tiles. The first tail comprises two small and one medium macro tile. The second tail instead is given by one big, two medium and two small macro tiles.

It is now possible to define for each class the z offset as the mutual point where the first tail ends and the second one starts. Another way to define z could be the point denoting where the first biggest macro tile of a certain region is placed within the

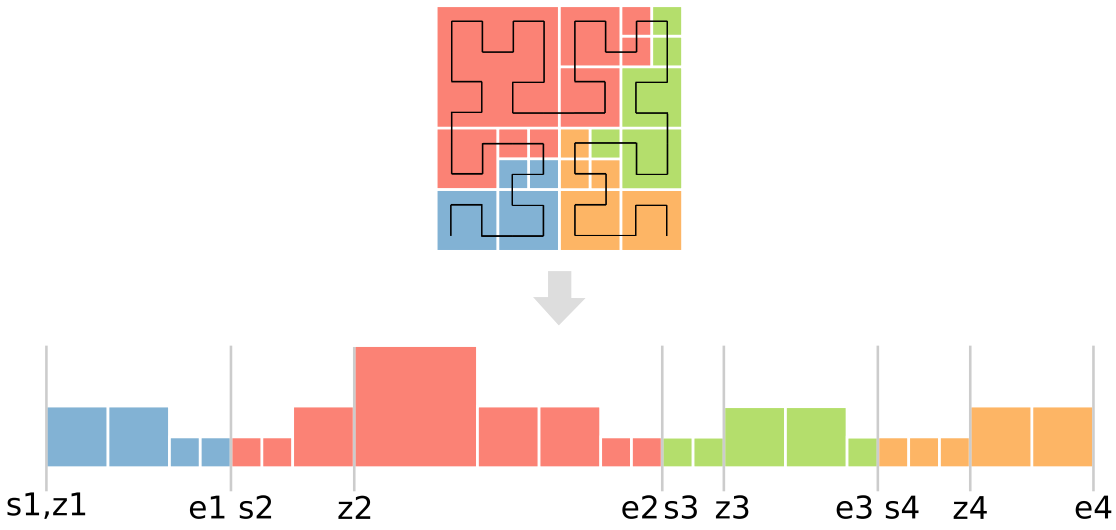


Figure 5.11: *The result of the spatialization step is shown as an unwrapped sequence of macro tiles where the offsets s, e and z are marked for each class. In particular z defines the point in which the biggest macro tile of a certain class starts within the sequence. In the case of the first blue class, the s and z offsets coincide.*

sequence.

The procedure for calculating it takes as input a class c and a logarithm base n. There are two main cases, the first one in which s and z corresponds and the second one in which they do not. The first case is the one regarding the first class within the unwrapped sequence of macro tiles shown in Figure 5.11 (The offset $s1$ and $z1$ of the blue class corresponds). Hence, the procedure provided below first return z equal to zero only when the value of s is zero. Considering the example shown in Figure 5.11 this happen only for $s1$ and $z1$. Otherwise, the algorithm loops until the first biggest macro tile is found. This can be computed by searching for the coordinates of the biggest base-n number, less than the end offset, and composed by a digit followed by all zeros.

```
get_z = (c, n) ->
  start_n = (c.start).toString(n)
  end_n = (c.end).toString(n)

  if start_n is '0'
    return parseInt(start_n, n)

  i = start_n.length - 1
  initial_i = i
  while start_n.length <= end_n.length
    new_start_n = replace_char start_n, i, '0'
    new_start_n = (parseInt('1' + Array(new_start_n.length - i + 1).
        join('0'), n) + parseInt(new_start_n, n)).toString(n)

    if parseInt(new_start_n, n) < c.end
      start_n = new_start_n
    else
```

```
      break

    if initial_i is start_n.length
      i -= 1
    else
      initial_i += 1

  return parseInt(start_n , n)
```

Base-n numbers calculation

Given the *order* and the offset s, e and z, the base-n numbers representing the coordinates of the macro tiles can be generated for each class in the input forest. As in the previous step, this phase exploits the two tails composing every class. The first tail ranges from s to z while the second one from z to e. The algorithm reported below calculates for both tails their ranges that are respectively the difference between z and s, and between e and z. The ranges, expressed as integers, are then converted to base-n. The resulting base-n ranges internally embrace within their digits the number of macro tiles that have to be generated and their size.

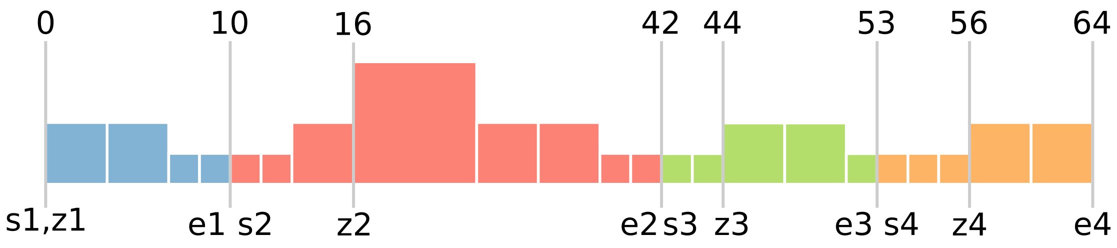


Figure 5.12: *This diagram shows a precise example of the sequence of macro tiles depicted in Figure 5.11. Offset values are reported on top of the gray vertical ticks while their corresponding offset labels are written on the bottom.*

We consider now the example shown in Figure 5.12. Each class is taken into account and their corresponding information are computed in the following:

- Blue class.
  - First tail: $z1 - s1 = 0 - 0 = (0)_{10} \rightarrow (0)_4$
    The difference between the offsets gives zero which converted to base-4 remains zero. This means that no macro tiles are generated for the first tale of the blue class. In fact, since $s1$ and $z1$ correspond to each other there is no range between them.
  - Second tail: $e1 - z1 = 10 - 0 = (10)_{10} \rightarrow (22)_4$
    The difference between the offsets gives $(10)_{10}$ which converted to base-4 gives 22_4. This value confirms the presence of 2 macro tiles of order 1 and 2 macro tiles of order 0.
- Red class.
  - First tail: $z2 - s2 = 16 - 10 = (6)_{10} \rightarrow (12)_4$
    The difference between the offsets gives $(6)_{10}$ which converted to base-4

gives $(12)_4$. This value confirms the presence of 2 macro tiles of order 0 and 1 macro tile of order 1.

- Second tail: $e2 - z2 = 42 - 16 = (26)_{10} \rightarrow (122)_4$
  The difference between the offsets gives $(26)_{10}$ which converted to base-4 gives $(122)_4$. This value confirms the presence of 1 macro tile of order 2, 2 macro tiles of order 1 and 2 macro tiles of order 0.

- Green class.
  - First tail: $z3 - s3 = 44 - 42 = (2)_{10} \rightarrow (2)_4$
    The difference between the offsets gives $(2)_{10}$ which converted to base-4 remains $(2)_4$. This value confirms the presence of 2 macro tiles of order 0.
  - Second tail: $e3 - z3 = 53 - 44 = (9)_{10} \rightarrow (21)_4$
    The difference between the offsets gives $(9)_{10}$ which converted to base-4 gives $(21)_4$. This value confirms the presence of 2 macro tiles of order 1 and 1 macro tile of order 0.
- Orange class.
  - First tail: $z4 - s4 = 56 - 53 = (3)_{10} \rightarrow (3)_4$
    The difference between the offsets gives $(3)_{10}$ which converted to base-4 remains $(3)_4$. This value confirms the presence of 3 macro tiles of order 0.
  - Second tail: $e4 - z4 = 64 - 56 = (8)_{10} \rightarrow (20)_4$
    The difference between the offsets gives $(8)_{10}$ which converted to base-4 gives $(20)_4$. This value confirms the presence of 2 macro tiles of order 1 and 0 macro tiles of order 0.

Finally, the base-n numbers corresponding to the coordinates of the macro tiles are computed using the offsets, order and range digits as shown below in the $get_m t$ and *get_mt_code* functions. In particular, the coordinate of a macro tile is computed in the *get_mt_code* function. First a base-n number bn is converted to base-4. Then a sequence of $order - bn.length$ zeros is prepended to bn. Finally the coordinate of the macro tile is returned as the substring of bn ranging from index 0 to index $order - digits$.

```
get_mt_code = (offset, order, digits) ->
  bn = offset.toString(4)
  bn = Array(order - bn.length + 1).join('0') + bn
  bn = bn[0...order-digits]
  return bn

get_mt = (c, order) ->
  macro_tiles = []

  ### Range from START to Z
  ###
  if c.start isnt c.z
    offset = c.start
    start_to_z = (c.z - c.start).toString(4).split('').reverse().join
        ('')
    for d,i in start_to_z
      for j in [0...parseInt(d)]
        digits = start_to_z.length - (start_to_z.length - i)

        mt = get_mt_code offset, order, digits
        macro_tiles.push mt

        offset += parseInt('1' + Array(digits + 1).join('0'), 4)

  ### Range from Z to END
  ###
  offset = c.z
  z_to_end = (c.end - c.z).toString(4)

  for d,i in z_to_end
    for j in [0...parseInt(d)]
      digits = z_to_end.length - i - 1

      mt = get_mt_code offset, order, digits
      macro_tiles.push mt

      offset += parseInt('1' + Array(z_to_end.length - i).join('0'),
         4)

  return macro_tiles
```

Macro tiles spatialization

This step specifically corresponds to the actual spatialization of the data since a position and a size are calculated for each macro tile. Every class is expressed as a set of base-n numbers that are taken as input. Since these values define the coordinates of the macro tiles in a non-cartesian coordinate system, a conversion is needed.

Every base-n number of a certain class is converted from a string of digits to a specific position in the space and a size that will be subsequently used for displacing and sizing macro tiles.

A layout function $mt\_function$ is used for computing the spatialization. It takes as input a $size$ value defining the display dimensions and a $mapping$ function that is responsible for coordinate conversion since it embodies the path of a certain space-filling curve.

```
mt_layout = (mapping, size) ->

  return (digits) ->
    m = mapping digits

    return {
      x: m.j/Math.pow(2,m.n)*size,
      y: m.i/Math.pow(2,m.n)*size,
      dx: 1/Math.pow(2,m.n)*size,
      dy: 1/Math.pow(2,m.n)*size,
      digits: m.digits
    }
```

It is therefore necessary to write a specific *mapping* function in order to spatialize data according to the particular features of a certain space-filling curve. For example, in the case of Hilbert the following mapping function can be used:

```
hilbert = (digits) ->
  n = digits.length
  l = 1
  i = 0
  j = 0

  for d in digits by -1
    switch d
      when '0'
        i_temp = i
        i = j
        j = i_temp
      when '1'
        i += l
      when '2'
        i += l
        j += l
      when '3'
        i_temp = i
        i = l - j - 1
        j = 2*l - i_temp - 1

    l = 2*l

  return {
    j: j,
    i: i,
    n: n,
    digits: digits
  }
```

The result of the spatialization is a converted version of macro tiles that can be mainly described by three attributes:

- *name*. It is the label of the macro tile.;
- *x and y*. They identify the position of the macro tile in a two-dimensional space;
- *size*. It depends on the kind of tile. For instance, square tiles need only a value specifying the length of their sides. Rectangular tiles requires two values for their sides. Hexagonal tiles are instead characterized by one value defining the side or the radius.

5.3.3 Merging

The merging step is responsible for creating the final regions of the map corresponding to the class nodes defined in the forest of a compound network. The spatialized macro tiles are taken as input and merged together forming the most deep level of regions of

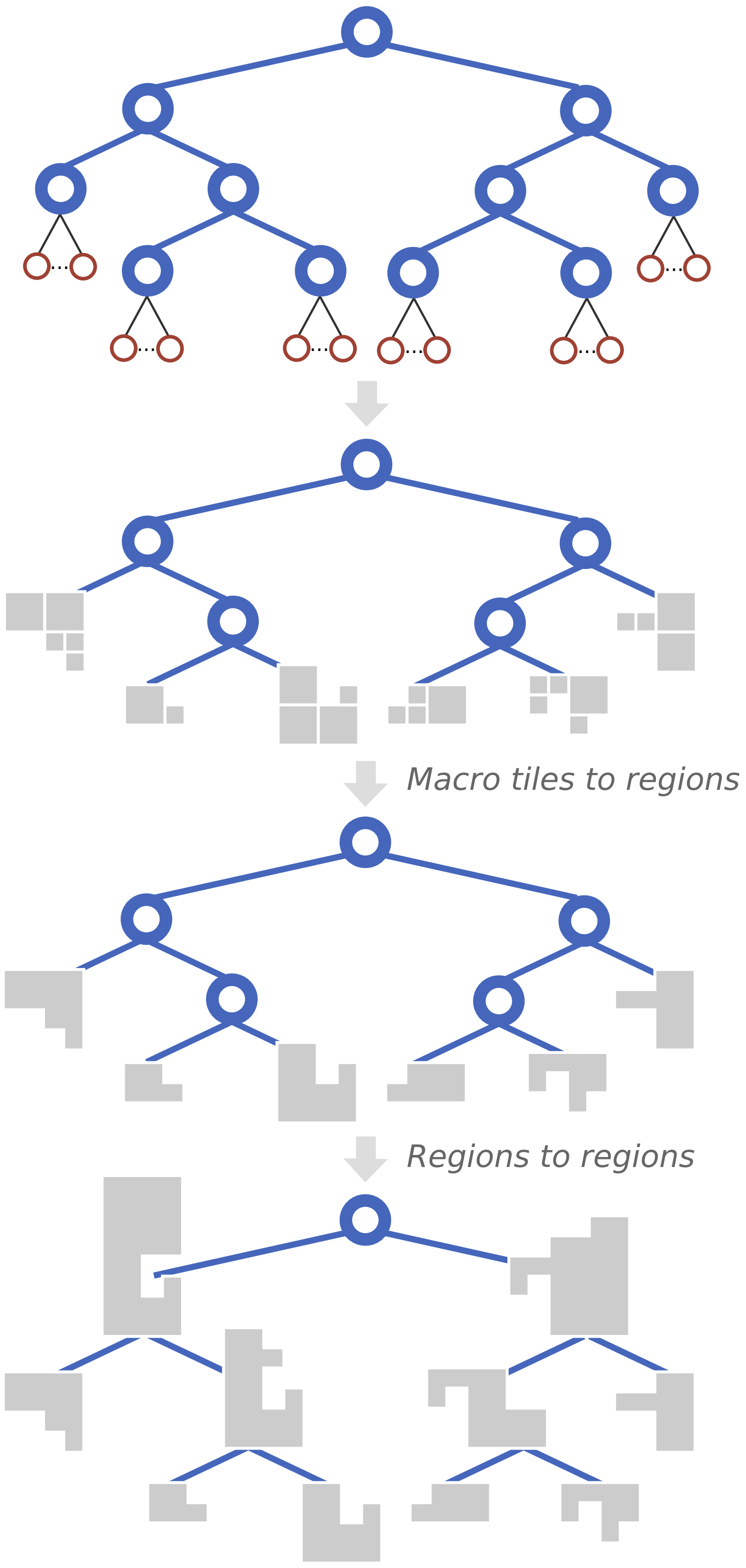


Figure 5.13: *The most specific regions are firstly computed by merging macro tiles. Then, by following the hierarchical relationships, bigger regions are constructed starting from the ones previously computed.*

the forest. Then additional merging operations are performed in order to produce also the regions corresponding to the class nodes positioned at an intermediate level in the forest.

As shown in Figure 5.13, two different kinds of merge operation are executed:

1. *Macro tiles to regions*. It is the first merge operation computed and it is performed only one time for each class in the forest. It computes the *most specific regions* corresponding to the deepest class nodes in the trees of the forest. Macro tiles, expressed by their parameters (i.e., x, y, size, digits), are taken as input and then merged according to the class they belong to;

2. *Regions to regions*. It is a recursive operation that is performed on every tree of the forest. It starts by merging the most specific regions previously generated. By following the hierarchical relationships, from the leaves of a tree to its root, regions are merged together forming bigger ones. The recursive procedure stops when it reaches the nodes directly connected to the root.

```
proc = (node) ->
  if node is root
    return
  else
    merge node.children
```

The final result of the merging process, shown on the bottom of Figure 5.13, is a new forest where the nodes of the trees are the regions of the map. The most wide and generic regions are located at the top of the hierarchy while the most specific one at the bottom.

5.4 Visual mapping transformation

The visual mapping transformation, specifically called *viewport filtering* in the case of our approach, is the last step of the visualization pipeline. It is responsible for transforming the map data into a map view which can be consulted by users.

First of all, it is essential to mention an important concept devised by Shneiderman called *Visual Information Seeking Mantra* that is a relevant guideline, very important to follow for the design and development of a visualization. The Mantra states *"Overview first, zoom and filter, then details on-demand"* [91] and means that the overview of a data set should be provided first in a visualization, then zooming and filtering mechanisms should be made available in order to focus the attention on a specific part of the data, specific details should be finally displayed only when demanded by users.

The viewport filtering presented in this section is a specific implementation of the Mantra that literally follows it. In fact, an interactive, geometric and semantic zoom has been used for implementing the zoom and filter operation of the Mantra. Moreover, this particular zoom behaviour is very natural and simple to use in the case of cartographic maps such as the one our approach constructs. It is a very common mechanism adopted by several applications for the consultation of traditional geographic maps. The term of viewport has to be intended as the map at a certain level of zoom displayed and visible on the screen of a computer monitor.

If properly implemented, the viewport filtering could really improve the performance of a visualization showing large amounts of data. In fact, instead of visually representing all the information describing a map, it only displays the ones that are necessary, in particular the ones users expect to see in the screen they are watching. There is no reason to spend costly effort for rendering all the data of the map when only the part zoomed by users is really necessary since it is the only one she can see. In the specific case of a map, the viewport filtering shows only the regions of the map the user has zoomed without rendering the others that can not be seen at the level of zoom reached. More precisely, only the regions of the map that appear in the screen will be visualized.

For instance, supposing you are using a common map online service and you have zoomed on a specific italian region such as Lombardy. There is no need of rendering other countries in Europe, or even other continents such Asia since they cannot be seen into the viewport the user is watching. This mechanism relies on the interactive nature of an application and is very effective since it leaves users unaware about the filtering.

The viewport filtering accepts as input the map data produced by the visualization transformation and shown at the bottom of Figure 5.13. This data structure is a forest where nodes are the class nodes of the compound network expressed by a set of attributes defining their position, size and shape. The links connecting them are the vocabulary links defined by the ontology. So, how can be such a structure transformed into a interactive map view?

According to the Mantra, the first visualization presented to the users should be an overview of the data set. For this reason, the most generic regions of the forest are firstly shown on the map. The logical separation of the trees of the forest is represented in the visualization as a spatial gap between the trees that are displayed as isolated regions. The resulting representation visually resembles an archipelago of islands. This first view gives a generic idea of the data set since it allow users to understand its general structure. Each island is then characterized by its own particular hierarchical structure that is initially depicted as the composition of its most generic regions, located on the top of their corresponding tree. The details about how islands are placed on the space will be given in Section 5.4.4.

It is important to specify that the terms forest and tree are used for referring to data structures while the terms archipelago and island for dealing with the visual representations of the data.

The overview is only the starting point for the users who can additionally explore the map by interacting with it as discussed in the following.

5.4.1 Semantic zoom

Once the map is ready and showing the overview, a zoom behaviour can be used by users for focusing their attention only on certain parts of the map and thus examining that specific portion of the data set. A particular feature called *semantic zoom* allows to explore and visit the hierarchical structure characterizing each island within the map.

Differently from a conventional geometric zoom where objects vary only in terms of their size, semantic zoom is a more complex kind of zoom mechanism that, beyond scaling sizes, could also change the visual representation of objects and their presence on the display by make them appearing and disappearing according to the zoom level

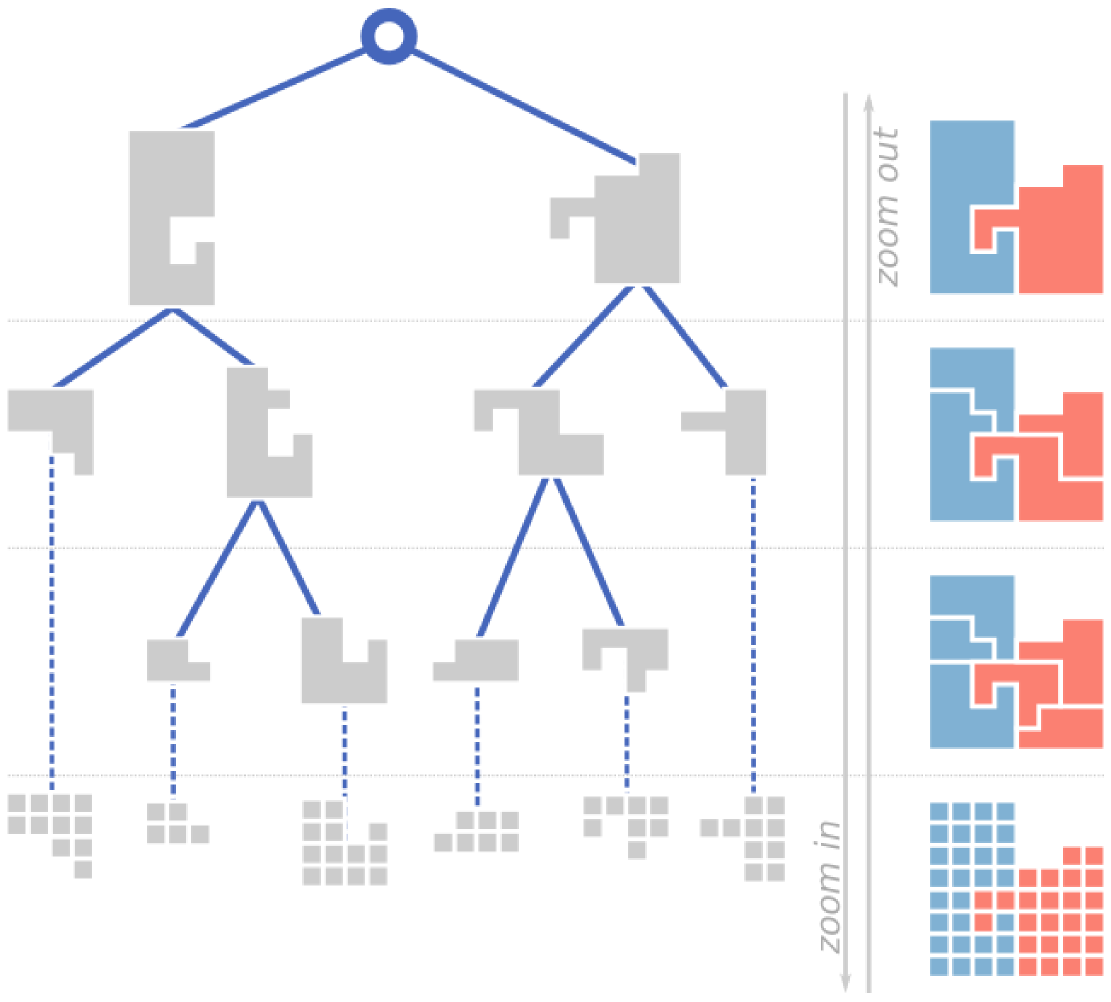


Figure 5.14: *On the left, the input is a hierarchical structure where nodes are regions that are displayed in the resulting visualizations on the right. Initially, the overview of the data set is provided as a visualization only composed by the regions at the top of the hierarchy. Then, the visualization is updated according to the level of zoom. By zooming in, generic regions are substituted by their children regions that provide a more specific view of the data set.*

reached [18, 73].

Once users perform a zoom-in operation, the initial overview of the data set is replaced by a smaller portion of the entire map corresponding to the zoomed part. At the same time, new content is loaded into the map for mainly characterizing it. More specifically, the generic regions composing the islands shown in the overview are exploded and replaced by their child regions that represent more specific classes of the hierarchies. Hence, from the user perspective, the act of zooming in corresponds to ask for a more specific and richer representation of a part of the map, while the act of zooming out identifies the need of having a more generic view. In terms of data structure, as shown by the diagram in Figure 5.14, the act of zoom-in and zoom-out respectively correspond to visit lower level of the regions hierarchy, from the root to the leaves, and vice versa.

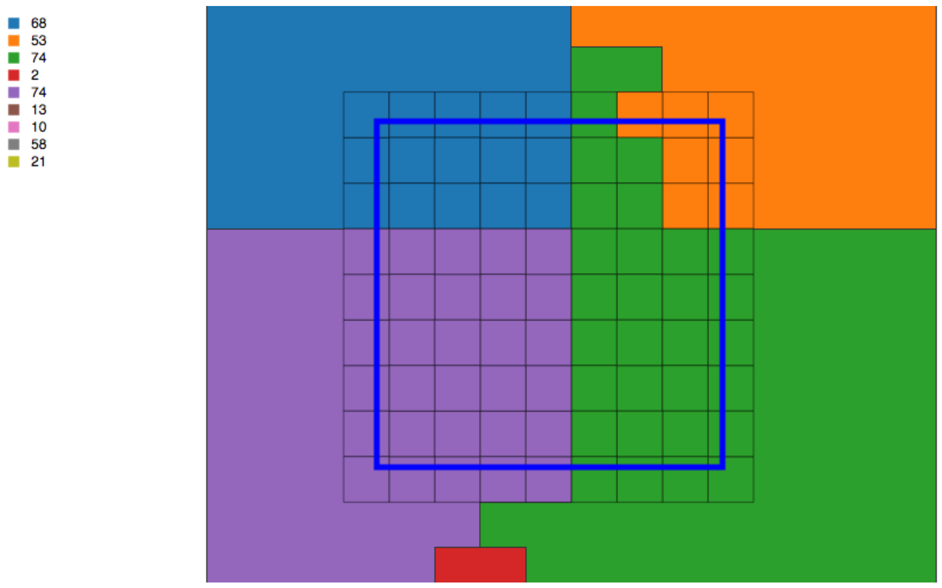


Figure 5.15: *An example showing how the viewport (blue square) works. Since loading all the tiles in a map could be quite costly, only the ones in the viewport are shown. In the real application, the viewport would fill all the screen available but still no tiles would be loaded outside of it. In this way, it is possible to give users the impression of having all tiles loaded even if only the ones framed are shown. An interactive version of this visualization can be consulted at http://bl.ocks.org/fabiovalse/bda5166279160e6ddab09f31e2b7a83d*

5.4.2 Level of detail

The *level of detail* represents the most specific level a map can provide. It is a view that can be obtained by reaching the maximum level of zoom allowed. Differently from the overview and the intermediate levels of zoom, where regions are displayed on the map, this particular view additionally decomposes regions in single tiles representing the instance nodes of the compound network. Once users reach a very high level of zoom, the level of details is automatically activated by the semantic zoom that explodes regions into single tiles.

The level of detail represents the conclusion of the user exploration process that starts with the overview showing the most generic parts of the data set, continues displaying more specific and enriched portions of it, and concludes with tiles that are the most specific piece of information the data set contains.

All these zooming operations, performed by users, are immediately processed by the system that smoothly updates the map giving the feeling of getting closer to something as happens with on-line map services. In order to provide a responsive user experience, the viewport filtering handles performance issues. For example, the tiles, that have to be loaded once the level of details is activated, must be filtered according to the portion of map displayed. This particular feature prevents the loading of large amounts of data that would make slower and less responsive the overall interaction. The example shown in Figure 5.15 simulates how it is possible to create such behaviour.

A blue viewport is placed in the center of the visualization and represents the screen typically consulted by users. By zooming and panning the map it is possible to imagine that tiles are only displayed within the viewport boundaries.

5.4.3 The English dictionary example

A simple example, whose interactive version is available on-line[3] and displayed in Figure 5.16, is provided in order to better clarify how the visualization works. The data set represented in the map is not a Linked Data one but the English Dictionary. Even if the example is not so relevant in term of content, since it is not encoded as Linked Data, it is useful for presenting how the approach works. The data set precisely counts 235,886 elements. It is therefore interesting to look at how the implementation of the map performs in term of map and content loading. The visualization takes around 20 seconds in order to display the map because all the operations (e.g., sorting, spatialization and merging) for creating it are performed on the client side. Furthermore, by performing these processes a pre-process step the time can be further decreased.

Moreover, the flat list of English words must be firstly transformed into a hierarchical structure in order to be compliant with our requirement. The prefixes of words, starting from single letters to prefixes composed by three letters, are used in order to build the hierarchy. The overview, as shown in Figure 5.16, is given by the regions corresponding to the alphabetical letters. By zooming in, for instance on the letter A region, its sub-regions, such as Ab, Ac and Ad, are automatically loaded. By additionally zooming in, it is possible to reach a point in which no more sub-regions are loaded but single tiles are shown as squares containing the label of the word they represent.

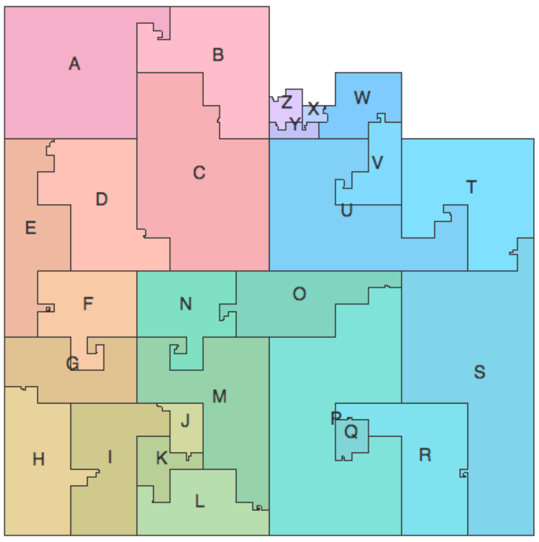
5.4.4 Islands Placement

As already introduced above, the particular forest data structure is visualized as an archipelago of islands representing the trees of the forest. In this section, the process for placing islands on a two-dimensional space is illustrated. The main idea is to exploit the force layout, typically used for representing node-link diagrams, for finding a disposition for the islands. Normally, this layout deals with nodes depicted as simple shapes such as circles or squares, or an image or icon.

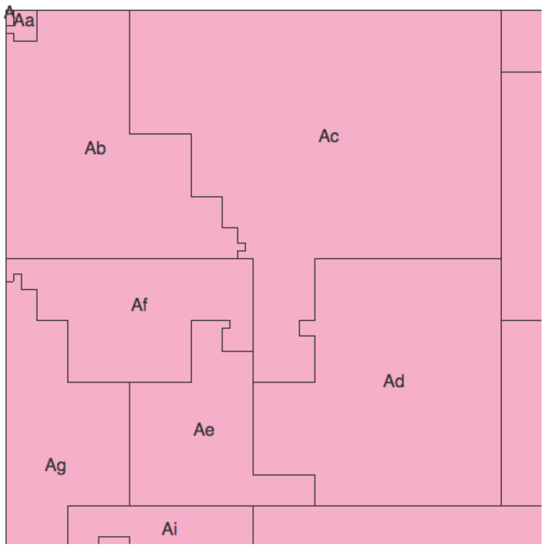
First, it is necessary to displace circles in a non-overlapping way. A collision detection algorithm, implemented by Mike Bostock[4], has been used in order to guarantee a defined padding between circles. The example, shown in Figure 5.17, implements this approach and also allows to change the padding distance between circles using the slider at the top. By using this algorithm, it is now simpler to place islands of simple circles in a non-overlapping way. As shown in Figure 5.18, islands are enclosed in circles that are in turn used as the input of the algorithm that is executed with padding equal to zero. In fact, since the islands are within the circles non overlapping can occur.

In order to prevent different displacements at every execution of the force-layout, the random seed is set to a fixed value in order to always produce the same configuration.

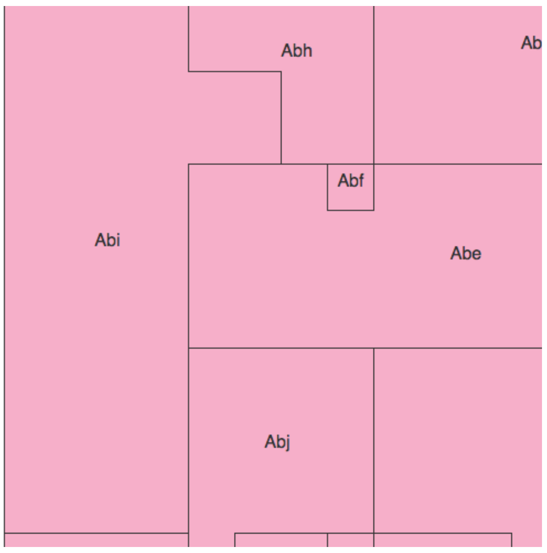
[3]`http://bl.ocks.org/fabiovalse/c5e0d866c4832264ff2ab75412558245`
[4]http://bl.ocks.org/mbostock/1747543



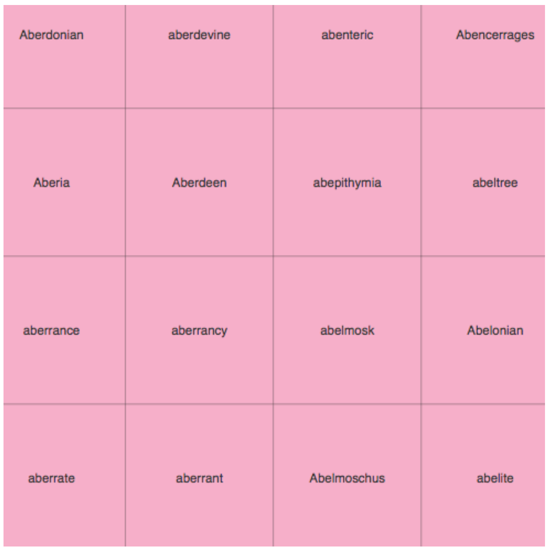
The overview of the map shows the most generic regions of the hierarchy corresponding to the letters of the alphabet.



By zooming to a specific part of the map, regions are exploded and replaced by their sub-regions that describe their sub-hierarchy.



By additionally zooming in, more specific sub-regions are loaded.



The deepest level of zoom shows the single elements of the data set. In this case the words of the english vocabulary are depicted as squared tiles.

Figure 5.16: *An interactive version of this visualization can be consulted at http://bl.ocks.org/fabiovalse/c5e0d866c4832264ff2ab75412558245*

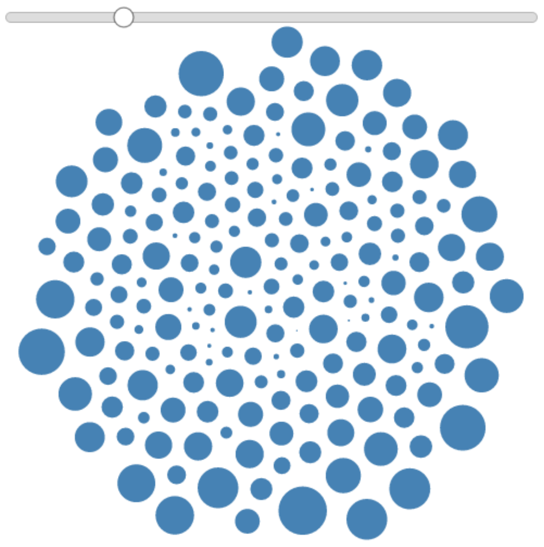


Figure 5.17: *An interactive version of this visualization is available at* `http://bl.ocks.org/fabiovalse/bf9c070d0fa6bab79d6a`.

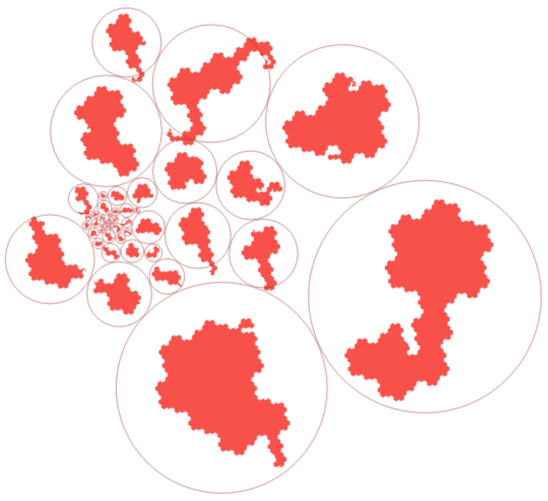


Figure 5.18: *An interactive version of this visualization is available at* `http://bl.ocks.org/fabiovalse/e7d2d1eba207b0a979ad`.

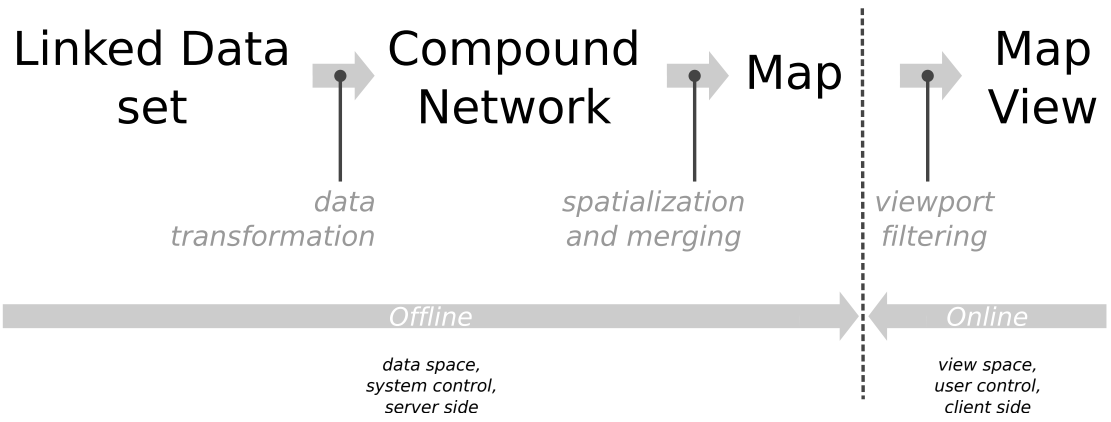

Linked Data set
Compound Network
Map
Map View
data transformation
spatialization and merging
viewport filtering
Offline
Online
data space, system control, server side
view space, user control, client side

Figure 5.19: *The visualization pipeline can be examined in terms of offline and online computation. The dashed vertical line defines the boundary between the two kinds of computation. The offline part corresponds to the data space, system control and server side of the application while the online part normally represents the view space, user control and client side.*

5.5 Offline vs. online

Given a series of operations needed for the construction of an application, it is interesting to analyze whether their computation has to be performed *offline* or *online*. On one hand offline data processing allows to avoid online execution effort, while on the other hand online computation permits, if desired, to be totally independent from any pre-processing procedure.

Offline and online computation define a distinct separation between different aspects characterizing an application.

- *data space and view space.* The offline computation defines the space of the data since it is responsible for different operations needed for preparing data such as transformations, conversions or translations. Online computation instead represents the view space since it aims to visualize data and provide their visual representations;
- *system control and user control.* The entity owning the control is another aspect characterizing offline and online computation. While in the former the control is kept by the system that performs its specific operations, in the latter users hold the control since they decide which are the data they are interesting in by interacting with the view;
- *server side and client side.* Since the visualization approach presented has been devised for being developed with Web technologies, a distinction is given by the execution location. Offline procedures are therefore executed on the server side while online computation on the client side.

In the specific case of our visualization pipeline, there is no fixed configuration establishing whether each step has to be executed offline or online. In fact, a high level of *flexibility* is available in order to decide, depending on the specific case, which is the best configuration. For instance, when large data sets have to be visualized, it is convenient to move almost the whole computation offline in order to prevent online costly effort. On the contrary, when a very frequently changing data set has to be handled, it is more advantageous to keep the computation online in order to quickly provide data variations.

The boundary between offline and online, as shown in Figure 5.19, can be moved to different points along the pipeline. The maximum point it can be shift on the right is immediately before the viewport filtering once the map data have been computed. It can be moved to the left even before the first step. Thus, while the offline part can be avoided, the online part is always necessary at least reduced to the viewport filtering.

CHAPTER 6

Case studies

SOME case studies are presented in this chapter in order to demonstrate the approach previously described. We choose two well-known Linked Data sets of the Semantic Web research field, DBpedia [7] and LinkedMDB [44], for testing and evaluating the approach. They are very different in terms of content, size and structure. DBpedia is a cross-domain data set, one of the biggest in the LOD cloud with almost 5 million resources and structured with a hierarchical ontology. LinkedMDB is a data set containing movie-related data, counting almost 700,000 resources and modelled with a ontology composed of a collection of disconnected classes.

In order to apply our approach, we implemented two web-based prototypes for DBpedia and LinkedMDB. The web applications mainly constitute four components:

1. *Map.* It firstly displays the overview of the data set showing all the instances and classes within it. Zoom and pan mechanisms allow users to filter out certain regions, move within the map and focus their attention on specific regions and instances. Initially, only the regions with a suitable size for displaying their label identifying their ontological class show it while the other region labels are automatically loaded during a zoom action. Clicking on an instance on the map triggers the loading of its properties both in the map itself and in the right-side infobox. All the instances connected to the selected one are highlighted on the map as red tiles linked by red lines. In the case of DBpedia, a metaphor for portraying instances as *cities* have been introduced, in order to support users to get orientation within the map and get a feel about the content of a certain region. Cities are visualized, as in traditional maps, as a point on a map with a label describing it.

2. *Infobox.* It comprises all the RDF triples in which a selected instance is involved. In particular, in order to help users in the consultation of data, classes, data proper-

ties, incoming and outgoing relations are organized and reported in a user-friendly way, by grouping objects according to the predicates and replacing URI prefixes labels. Furthermore, by clicking on an outgoing or incoming property, the relative content is loaded both on the map and in the infobox, allowing the data set navigation.

3. *Search Box.* The search functionality allows to perform a text search for a specific instance, show it on the map along with its connections, and load its properties in the infobox.

4. *Layers menu.* Multiple aspects of a data set can be displayed on top of the main visualization as additional thematic maps. Each region may be colored to represent for example: the depth of the corresponding class in the forest, the density (i.e., a normalization of the number of instances) of RDF triples, the density of the object properties, or the density of data properties describing the instances. The layers menu, on the top left, allows to switch between the different thematic maps available.

In the following, the two application prototypes of DBpedia and LinkedMDB will be firstly described and then the results of their evaluation will be presented. Both visual representations reflect the respective data set structure. The tree of DBpedia, rooted on the class “*Thing*”, is represented by one island composed by several regions, themselves partitioned in many hierarchical sub-divisions. An additional island has been added for containing instances without a class [78]. Instead, the disconnected classes of LinkedMDB produce a completely fragmented representation displayed as an archipelago of islands, since no root is defined in the ontology. Moreover, it is possible to notice that, differently from DBpedia, the regions of LinkedMDB have no sub-division.

In order to demonstrate the wide applicability of the approach, on common data sources not compliant with Linked Data principles and unrelated to Semantic Web, an additional use case constructed with the NCBI taxonomy database, is illustrated.

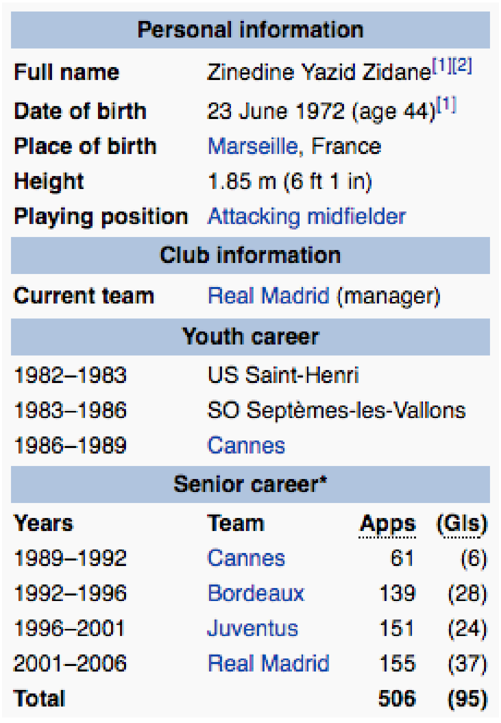
Finally, an experimental study for displacing resources according to a similarity measure is presented by showing some preliminary examples.

6.1 DBpedia

DBpedia is one of the most important and successful Linked Data sets within the Semantic Web. It has a significant size, in fact it contains almost 5 million resources within the 2014 dump and it presents a hierarchical structure since its ontology counts around 700 classes. Figure 6.2 shows a screenshot of the application we developed and available at `http://wafi.iit.cnr.it/lod/dbpedia/atlas`. The largest classes in the data set (e.g., *Agent*, *Place*, *Work* and *Species*) can be easily identified by the initial overview. Another insight regards the *Career Station* class that is one of the largest, but its content appears initially quite unusual. By analyzing it, it is possible to discover that it is totally flat since it has no subclasses and it contains resources representing the step in the career of people, in particular athletes.

Moreover, by further investigating on these resources it is possible to notice that they are very peculiar since they are actually the result of the conversion of specific

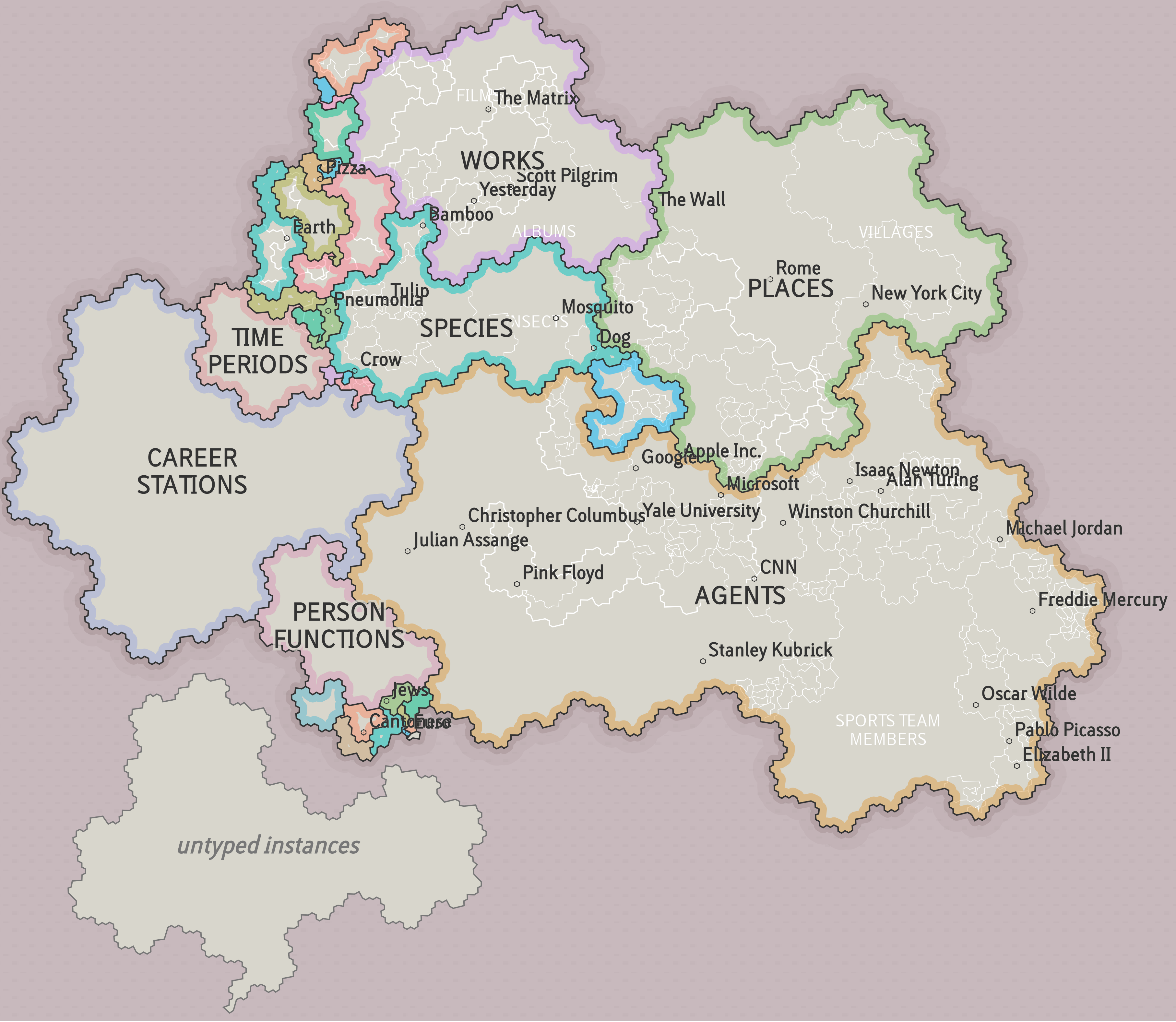
Wikipedia relationships into DBpedia. For instance, considering the resource of the Zinedine Zidane soccer player[1], it is possible to find ten RDF triples having *hasCareerStation* as predicate and as object an instance of the class CareerStation. These CareerStation resources have a URI such as *http://dbpedia.org/resource/Zinedine_Zidane__5*, *.../Zinedine_Zidane__6*, *.../Zinedine_Zidane__7* and so on. The interesting aspect is that, differently from the other resources of DBpedia, there exist no Wikipedia article corresponding to these resources. However, by looking at the Wikipedia article of Zinedine Zidane[2] we can find in the infobox (Figure 6.1) on the top right of the page the information from which the CareerStation resources have been created. In fact, by looking at the section labeled as *Senior career* it is possible to see the relationships of the soccer player with the team he played in the past with some information such as the time periods, number of appearances and goals. It is now clear that the reason of describing the relationships, connecting a soccer player to a team, as a resource of DBpedia has been driven by the fact that RDF does not allow to specify attributes on predicates. This means that RDF does not enable to simply define a relationship between a player and a team using a predicate, such as *playedFor*, having one or more attributes associated. Instead, it is necessary to define a new resource representing the step in the career of a player in a precise period of time, for a specific team, playing a certain amount of matches and scoring a certain amount of goals.

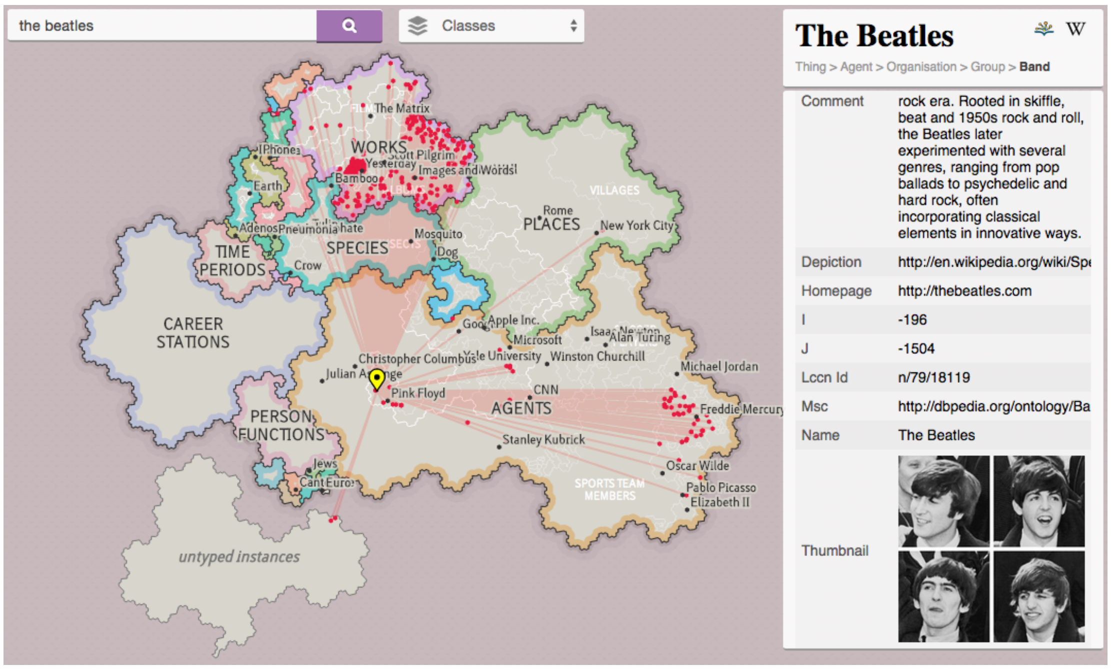


Personal information			
Full name	Zinedine Yazid Zidane[1][2]		
Date of birth	23 June 1972 (age 44)[1]		
Place of birth	Marseille, France		
Height	1.85 m (6 ft 1 in)		
Playing position	Attacking midfielder		
Club information			
Current team	Real Madrid (manager)		
Youth career			
1982–1983	US Saint-Henri		
1983–1986	SO Septèmes-les-Vallons		
1986–1989	Cannes		
Senior career*			
Years	**Team**	**Apps**	**(Gls)**
1989–1992	Cannes	61	(6)
1992–1996	Bordeaux	139	(28)
1996–2001	Juventus	151	(24)
2001–2006	Real Madrid	155	(37)
Total		**506**	**(95)**

Figure 6.1: *A portion of the infobox of the Wikipedia article about the soccer player Zinedine Zidane. It is interesting to look at the information under the* Senior career *section. In fact, each row of the table reported in this section, such as "1992–1996 Bordeaux 139 (28)", has been translated by DBpedia in a resource like* `http://dbpedia.org/page/Zinedine_Zidane__5`.

[1] `http://wafi.iit.cnr.it/lod/dbpedia/atlas/#Zinedine_Zidane`
[2] `https://en.wikipedia.org/wiki/Zinedine_Zidane`


The Matrix
FILMS
WORKS
Scott Pilgrim
Yesterday
Bamboo
ALBUMS
Pizza
Earth
The Wall
VILLAGES
Rome
PLACES
New York City
Tulip
Pneumonia
SPECIES
INSECTS
Mosquito
Dog
TIME
PERIODS
Crow
CAREER
STATIONS
Google
Apple Inc.
Microsoft
Isaac Newton
Alan Turing
SOCCER
Christopher Columbus
Yale University
Winston Churchill
Julian Assange
Michael Jordan
Pink Floyd
CNN
AGENTS
Freddie Mercury
PERSON
FUNCTIONS
Stanley Kubrick
Jews
Oscar Wilde
Cantonese
SPORTS TEAM
MEMBERS
Pablo Picasso
Elizabeth II
untyped instances



The Beatles

Thing > Agent > Organisation > Group > **Band**

Comment	rock era. Rooted in skiffle, beat and 1950s rock and roll, the Beatles later experimented with several genres, ranging from pop ballads to psychedelic and hard rock, often incorporating classical elements in innovative ways.
Depiction	http://en.wikipedia.org/wiki/Sp
Homepage	http://thebeatles.com
I	-196
J	-1504
Lccn Id	n/79/18119
Msc	http://dbpedia.org/ontology/Ba
Name	The Beatles
Thumbnail	

Figure 6.2: *A screenshot of the DBpedia application, available online at* `http://wafi.iit.cnr.it/lod/dbpedia/atlas`*. The code is open source and hosted on GitHub (*`https://github.com/fabiovalse/dbpedia_atlas`*). The map, composed by a main island and a smaller one containing the untyped instances, reflects the hierarchical structure of the ontological tree of DBpedia.*

By interactively zooming in and out of the map, performing searches and looking at how resources are connected, it is possible to find more interesting patterns. For instance, as shown in Figure 6.3, 6.4 and 6.5, it is possible to compare three resources belonging to the class *Company*: Google[3], Apple[4] and Microsoft[5]. By looking at the distribution of the entities connected to these companies we can see that Google is connected to a large amount of *Website* resources and a considerable amount of *Software* resources. Apple is connected only to a few *Website* resources and a large number of *Software* resources. Microsoft appears to be more similar to Apple than to Google.

In Figure 6.2, the resource `http://dbpedia.org/resource/The_Beatles` is currently selected and it can be localized on the map by the yellow placemark. The red links starting from it represent the relationships of The Beatles with other resources in DBpedia. Most of the links going towards the top of the map, are directed to the *Works* region where the songs and the albums of the band are located. The second biggest flow pointing to the bottom right side of the map, are directed to the *Singer* region, as shown by the city labeled as Freddy Mercury.

[3]`http://dbpedia.org/resource/Google`
[4]`http://dbpedia.org/resource/Apple_Inc.`
[5]`http://dbpedia.org/resource/Microsoft`

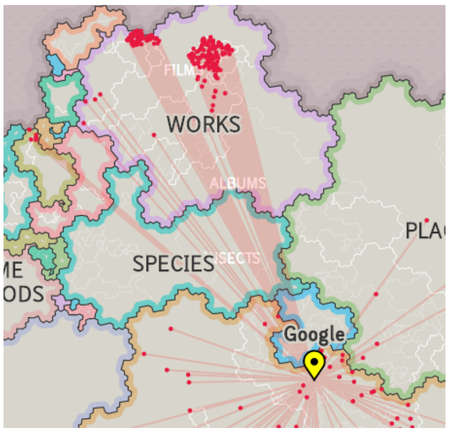


Figure 6.3: *The distribution of instances (red dots) connected to the entity* Google *(yellow placemark). A large number of dots gathers in the* Website *region (top left) and in the* Software *region (top middle).*

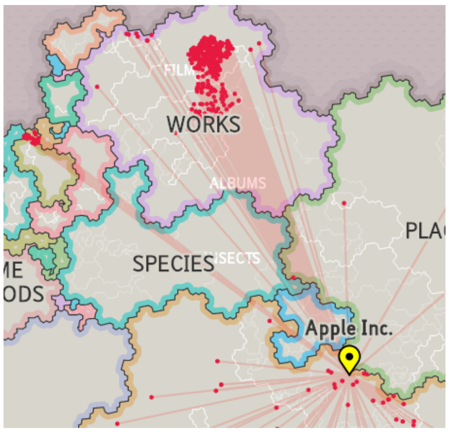


Figure 6.4: *When* Apple Inc. *is selected, the amount of websites decreases significantly, while* Software *becomes much more prominent. This is especially true for the lowest part of the region (*Video Game*). An interesting conglomerate appears on the left (*Device*).*

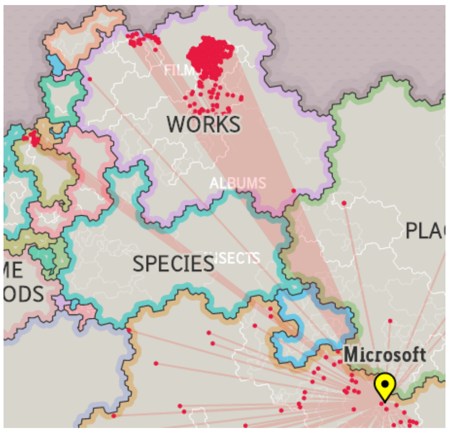


Figure 6.5: *The distribution for the instance* Microsoft *is more similar to the one for* Apple Inc. *than it is for* Google. *However, with regards to both* Device *and* Website*, it seems that Microsoft falls somewhere in between the other two.*

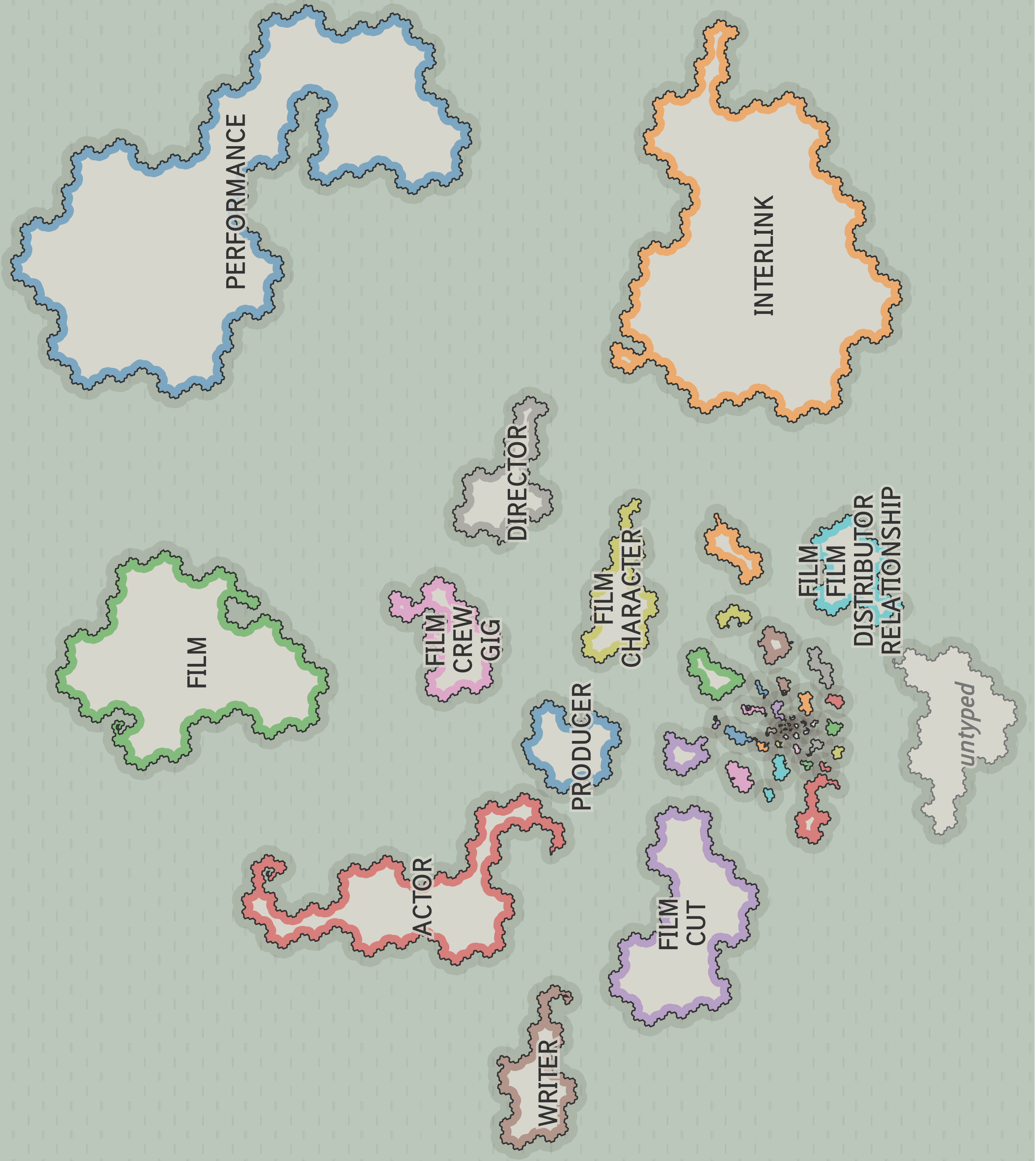
6.2 LinkedMDB

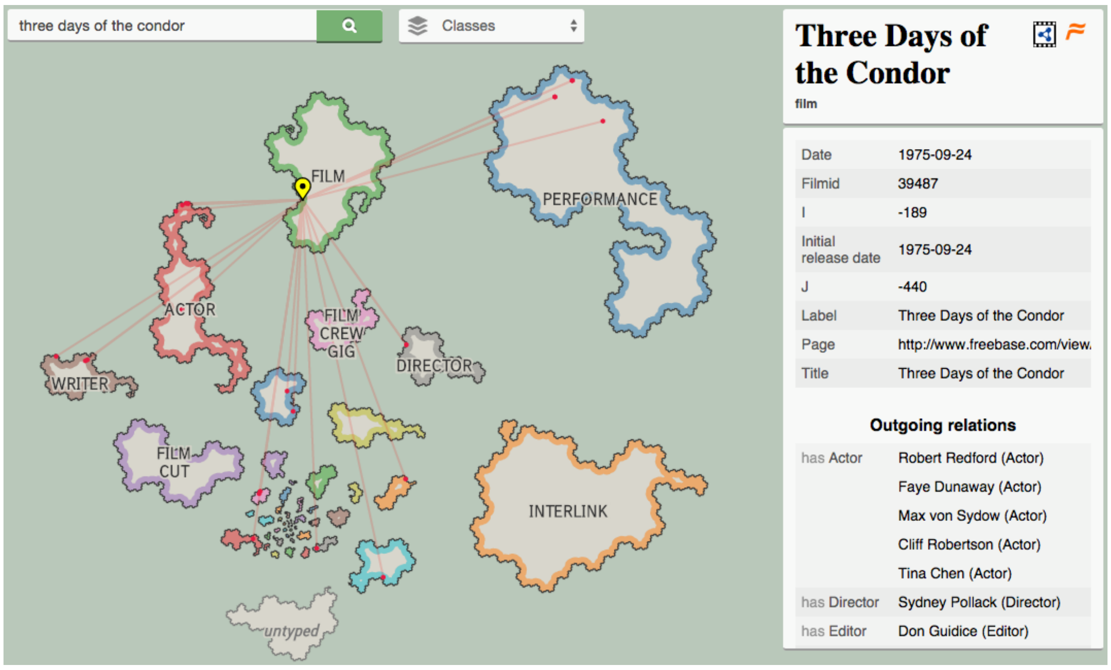
LinkedMDB is another important initiative within the LOD cloud that is derived from the popular online Internet Movie Database[6] (IMDb) concerning films, television programs, cast, production crew, fictional characters, their biographies and so on. As shown in Figure 6.6, LinkedMDB has a very different structure compared to the one of DBpedia. The map seems an archipelago of islands generated from the peculiar ontology the data set owns. The main reason behind this particular configuration is that no hierarchical structure has been adopted for LinkedMDB but only a disconnected list of ontological classes. This can be noted in the prototype, by interactively zooming on the map, and seeing that no sub-regions appears.

In Figure 6.6, the resource `http://dbpedia.org/resource/Three_Days_of_the_Condor` is currently selected and it can be localized on the map by the yellow place mark. The red links starting from it represent the relationships of the movie with other resources in LinkedMDB. Different kind of resources are connected to it: *Performance*, *Actor*, *Writer*, *Director* are just the major ones. By zooming into the map also it is possible to discover other relationships with resources belonging to the tiniest islands corresponding to the classes *Editor*, *Film Location* and *Film Subject*.

Among the insights that can be discovered within the data set it is interesting to understand which is the role of the *Performance* region since it is one of the biggest. Their elements basically represent the connection between a certain actor/actress and the character he/she played in a specific movie. Knowing this allows to find which are the movies in which a certain character has been played. For instance by searching for Neo it is possible to see different performance of Keanu Reeves in the Matrix movies. Otherwise, searching for the character as “Batman" allows to see the different performances made by distinct actors (e.g., Christian Bale, George Clooney) in the many movies shoot on Batman.

[6]`http://www.imdb.com/`


PERFORMANCE
INTERLINK
FILM
DIRECTOR
FILM CREW GIG
FILM CHARACTER
FILM FILM DISTRIBUTOR RELATIONSHIP
untyped
PRODUCER
ACTOR
FILM CUT
WRITER



Three Days of the Condor

film

Date	1975-09-24
Filmid	39487
I	-189
Initial release date	1975-09-24
J	-440
Label	Three Days of the Condor
Page	http://www.freebase.com/view.
Title	Three Days of the Condor

Outgoing relations

has Actor	Robert Redford (Actor)
	Faye Dunaway (Actor)
	Max von Sydow (Actor)
	Cliff Robertson (Actor)
	Tina Chen (Actor)
has Director	Sydney Pollack (Director)
has Editor	Don Guidice (Editor)

Figure 6.6: *A screenshot of the LinkedMDB application, available online at* `http://wafi.iit.cnr.it/lod/linkedmdb/atlas`*. The map, composed by an archipelago of islands, depicts the structure of the disconnected classes forming the ontology of LinkedMDB.*

6.3 Evaluation

We carried out a user evaluation with 19 users, collecting both qualitative and quantitative data. Since the intended users are people wanting to approach or to better understand a Linked Data set, all the testers we chose have a computer science background (no lay users were recruited). The 19 participants were 12 males and 7 females and they had different age ranges: 5 were between 20 and 30, 7 between 30 and 40 and 7 between 40 and 55.

Half of the people were presented with the DBpedia application, while the other half the LinkedMDB one. At the beginning of the session with each user, we asked them to read a short description about the dataset. For instance, in the case of DBpedia we used the following description:

DBpedia is a large collection of structured data automatically extracted from Wikipedia. It contains a lot of resources (e.g. "Galileo Galilei", "Pisa", "The Beatles", "Divine Comedy") organized in classes (e.g. "Person", "City", "Band", "Book"). Classes are connected together, forming a hierarchical structure from generic to specific. For example, "FootballPlayer" is a more specific class than "Athlete", which itself is more specific than "Person". Resources are also connected to each other (e.g. "Pisa" is the place of birth of "Galileo Galilei").

Then, we gave them some time (5 to 10 minutes) for freely interacting with the application. Finally, they had to fill out a questionnaire. To answer some of the questions they had to perform some tasks using the application. The questionnaire has been structured in four distinct parts for evaluating different aspects of the visualizations and the applications.

1. *Free interaction*. Three questions were defined for verifying whether the main features (e.g., zoom and pan, search, and click on the map) of the interface had been discovered and used by testers during the initial free interaction phase.

2. *Tasks*. Eleven tasks have been presented to users for testing if the application is useful for: i) visually identifying which are the main regions within a map; ii) looking up and locating resources; iii) recognizing which are the connections between resources, and iv) understanding how resources are categorized.

   After they finished each task, users were asked to state whether it was easy to solve. We adopted a scale of 5 balanced values (i.e., really difficult, difficult, neither easy nor difficult, easy and really easy) ranging from 1 to 5 for scoring the results of each task.

3. *Comprehension*. Three questions were asked for measuring whether the users had grasped the main aspects of the visualizations (e.g., the meaning of regions and their size).

4. *Final comments*. Four questions were finally asked for understanding whether the users thought the application was aesthetically pleasing, easy to use, useful and/or self-explanatory. We used a Likert scale [64] ranging from 1 to 5 for scoring the answers of the users about the aforementioned properties.

6.3.1 Results

The results of the test show that in the case of DBpedia 80% of the users autonomously discovered they could use the search feature and click on the map, while 70% the zoom and pan behaviour. In the case of LinkedMDB, all the users discovered they could click on the map, 88% used the search function and 75% of them used the zoom and pan mechanism. By taking into account those results, it is reasonable to think that the application requires a clearer way of showing that the zoom and pan feature is available. The plus/minus zoom buttons often included in interactive web maps could be a solution.

The study produced very promising results considering the percentages of correct answers and the average scores (Table 6.1 and 6.2) and also gave directions for improving the strength of the application in communicating the data set characteristics. The mean score of every task is above average for both DBpedia and LinkedMDB. One of the main aspects that came out from the study is the need for a closer interaction between the visualization of the map and the infobox. In fact, by observing the users when executing the tasks, we noticed that some are more inclined to navigate the map, while others prefer to just read the infobox.

All the users understood that the size of a region is determined by the amount of resources contained within it, while only a few of them really perceived the complexity of its hierarchical sub-structure. Almost all the users agreed on the aesthetic, ease and usefulness of the application while only a part of them think that it is self-explanatory, suggesting that a more intuitive user interface can be envisaged. A user suggested to add an introductory tutorial for explaining the main features the first time the application is accessed.

	Questions	%	Avg
Free Interaction	Have you used the pan & zoom feature?	70%	-
	Have you used the search feature?	80%	-
	Have you clicked on the map?	80%	-
Tasks	Which is the biggest class in the dataset?	90%	3.6
	Which are the main classes? Try to explain your answer.	100%	4.0
	Where is the resource “The Matrix"?	100%	4.2
	Which are the resources connected to “The Matrix"?	100%	4.2
	Where is the resource “The Beatles"?	100%	4.2
	Which are the resources connected to “The Beatles"?	100%	4.3
	Is “Alan Turing" directly connected to “Noam Chomsky"?	90%	3.7
	Is “Divine Comedy" directly connected to “Dante Alighieri"?	90%	3.5
	Is “Earth" directly connected to a resource classified as “Place"?	70%	3.4
	Is “Crow" directly connected to a resource classified as “Food"?	80%	3.9
	Which are the classes of the resource “Danube"?	100%	3.7
Comprehension	Some regions are bigger than others because: 1) they contain more resources; 2) they contain more classes; 3) they are deeper in the hierarchy; 4) they have more connections.	100%	-
	Does region “Agent" seem more complex than “Place"? 1) Yes, “Agent" is more complex; 2) No, “Place" is more complex; 3) They are of similar complexity; 4) I don’t know.	20%	-
	Does region “CareerStation" seem more complex than “Place"? 1) Yes, “CareerStation" is more complex; 2) No, “Place" is more complex; 3) They are of similar complexity; 4) I don’t know.	90%	-
Final Comments	The map is aesthetically pleasing	-	4.1
	The map is easy to use	-	3.4
	The map is useful	-	3.6
	The interface is self-explanatory	-	3.0

Table 6.1: *The overall result of each question of the survey about DBpedia grouped by category. The percentage of correct answers and the average score reported by users are shown in the last two columns. Scores range on a scale from 1 to 5 (from* really difficult *to* really easy *in the case of the tasks, while from* totally disagree *to* totally agree *in the case of the final comments).*

	Questions	%	Avg
Free Interaction	Have you used the pan & zoom feature?	78%	-
	Have you used the search feature?	89%	-
	Have you clicked on the map?	100%	-
Tasks	Which is the biggest class in the dataset?	88%	3.7
	Which are the main classes?	67%	3.8
	Where is the resource “The Matrix"?	100%	4.6
	Which are the resources connected to “The Matrix"?	78%	3.9
	Where is the resource “Sidney Lumet" (the film director)?	89%	4.3
	Which are the resources connected to “Sidney Lumet" (the film director)?	77%	4.4
	Is “Blade Runner" directly connected to “Rutger Hauer"?	100%	3.8
	Is “Marcello Mastroianni" directly connected to “Claudia Cardinale"?	78%	3.2
	Is “Christmas (Film Subject)" directly connected to a resource classified as “Actor"?	100%	3.9
Comprehension	Some regions are bigger than others because: 1) they contain more resources; 2) they contain more classes; 3) they are deeper in the hierarchy 4) they have more connections.	89%	-
	Does region “Film" seem more complex than “Actor"? 1) Yes, “Film" is more complex; 2) No, “Actor" is more complex; 3) They are of similar complexity; 4) I don’t know.	44%	-
	Does region “Performance" seem more complex than “Film"? 1) Yes, “Performance" is more complex; 2) No, “Film" is more complex; 3) They are of similar complexity; 4) I don’t know.	22%	-
Final Comments	The map is aesthetically pleasing	-	3.8
	The map is easy to use	-	3.6
	The map is useful	-	3.9
	The interface is self-explanatory	-	2.9

Table 6.2: *The overall result of each question of the survey about LinkedMDB grouped by category. The percentage of correct answers and the average score reported by users are shown in the last two columns. Scores range on a scale from 1 to 5 (from* really difficult *to* really easy *in the case of the tasks, while from* totally disagree *to* totally agree *in the case of the final comments).*

CHAPTER 7

Experimental studies

This chapter presents some preliminary studies that, in future developments, could be included in the main approach described in chapter 5. As already performed within atlases, thematic maps can be used for showing additional information about a certain geographic area. For this reason, examples of thematics maps applied to the ones produced by our approach are presented. Then, in order to show how flexible the approach presented in this thesis is, a prototype developed using data not coming from a Linked Data set is described. Finally, a study about the arrangements of elements on a surface based on their similarity is illustrated.

7.1 Thematic Maps

In this section the specific concept of *thematic map* devised for the approach presented in this thesis is illustrated. Traditional thematic maps are defined by Wikipedia as *a type of map especially designed to show a particular theme connected with a specific geographic area. These maps can portray physical, social, political, cultural, economic, sociological, agricultural, or any other aspects of a city, state, region, nation, or continent*[1].

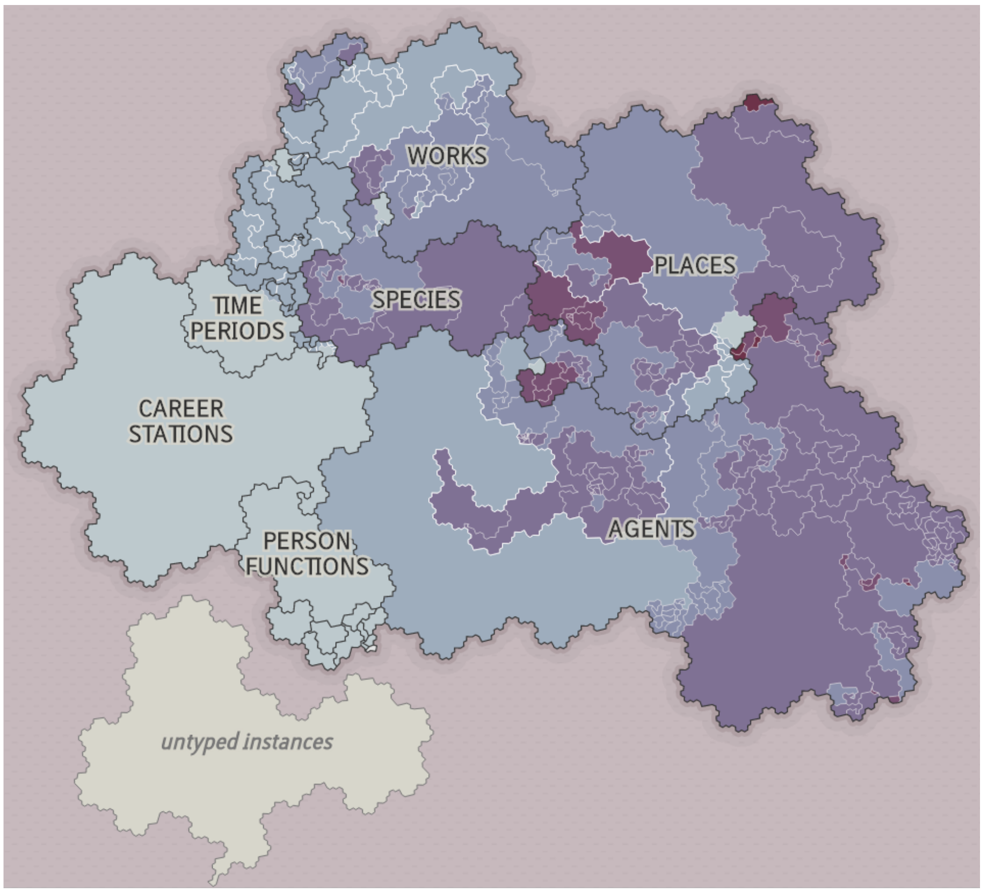
The idea of thematic map presented here is very similar. A thematic map is an additional map that can be used in order to show supplementary characteristics and features of a data set. In the case of Linked Data sets there are several aspects that can be displayed such as the classes categorizing instances or the predicates connecting them. Two examples have been implemented on the DBpedia and LinkedMDB prototypes.

The first one regards the level of depth of the classes in the ontology they belong. It is

[1] http://libweb5.princeton.edu/visual_materials/maps/websites/thematic-maps/firstxthenynowz.html

possible to show by colouring regions with a sequential colour scheme where the darker the colour the deeper the level of a class is. By looking at the result of this procedure in Figure 7.1, it is easy to understand which is the structure of the two data sets. Different colours characterize the thematic map of DBpedia and denote its hierarchical structure, shallow (e.g., *TimePeriod*, *CareerStation* and *PersonFunction*) and deep regions (e.g., the sub-regions of *Agent* and *Place*) are both present in the map. On the contrary, all the islands of LinkedMDB are coloured in gray due to the flat structure of the data set presenting no hierarchical level.

The second kind of thematic map, shown in Figure 7.2, is related to the density of object properties characterizing the instances belonging each class of the ontology. As before each class is coloured according to a sequential colour scale. The darker the colour the more dense the region. In the case of DBpedia, the overall map is quite similar to the previous one since the bottom-left side of the map is lighter denoting a less dense situation than the top-right side where the most dense classes can be found (i.e., *Soccer manager*, *Horse trainer* and *Jockey*). Differently from the previous flat map, LinkedMDB has a different situation where there are not so dense classes like *Performance*, quite dense ones such as *Interlink* and *Actor* and *Film* as the most dense one.



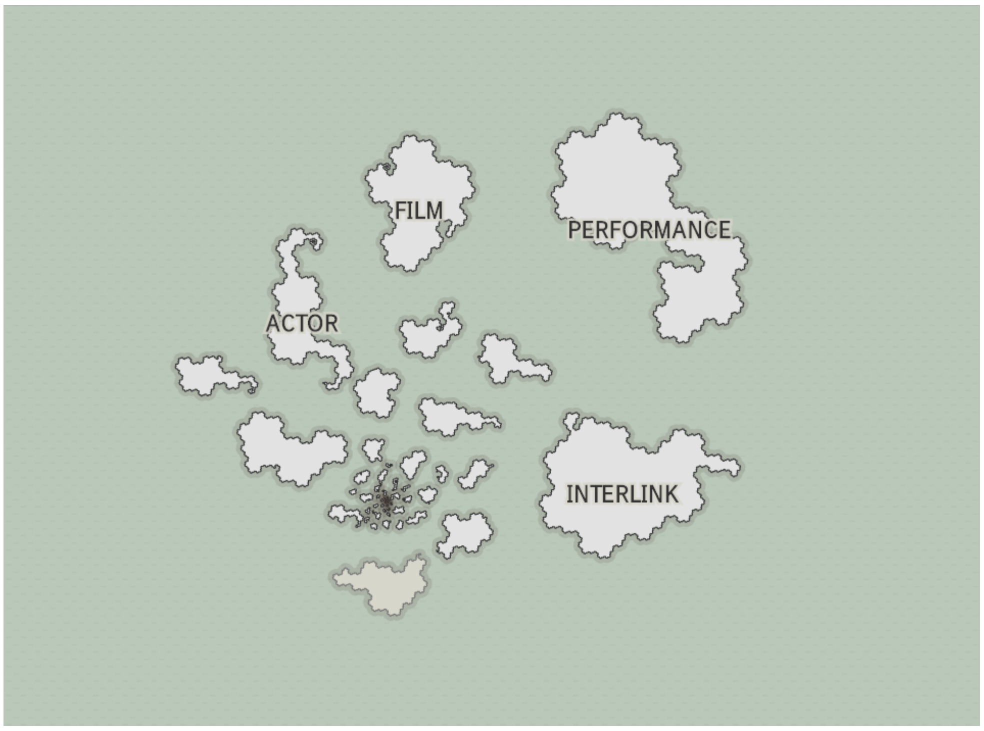


Figure 7.1: *Thematic maps of DBpedia and LinkedMDB, showing the depth of the classes in their ontology (the darker, the deeper). DBpedia has an island composed by several hierarchical levels, while LinkedMDB has an archipelago of totally flat islands. By inspecting the map of DBpedia, it can be seen that the deepest level of the ontology corresponds to the small Diocese class (top right).*

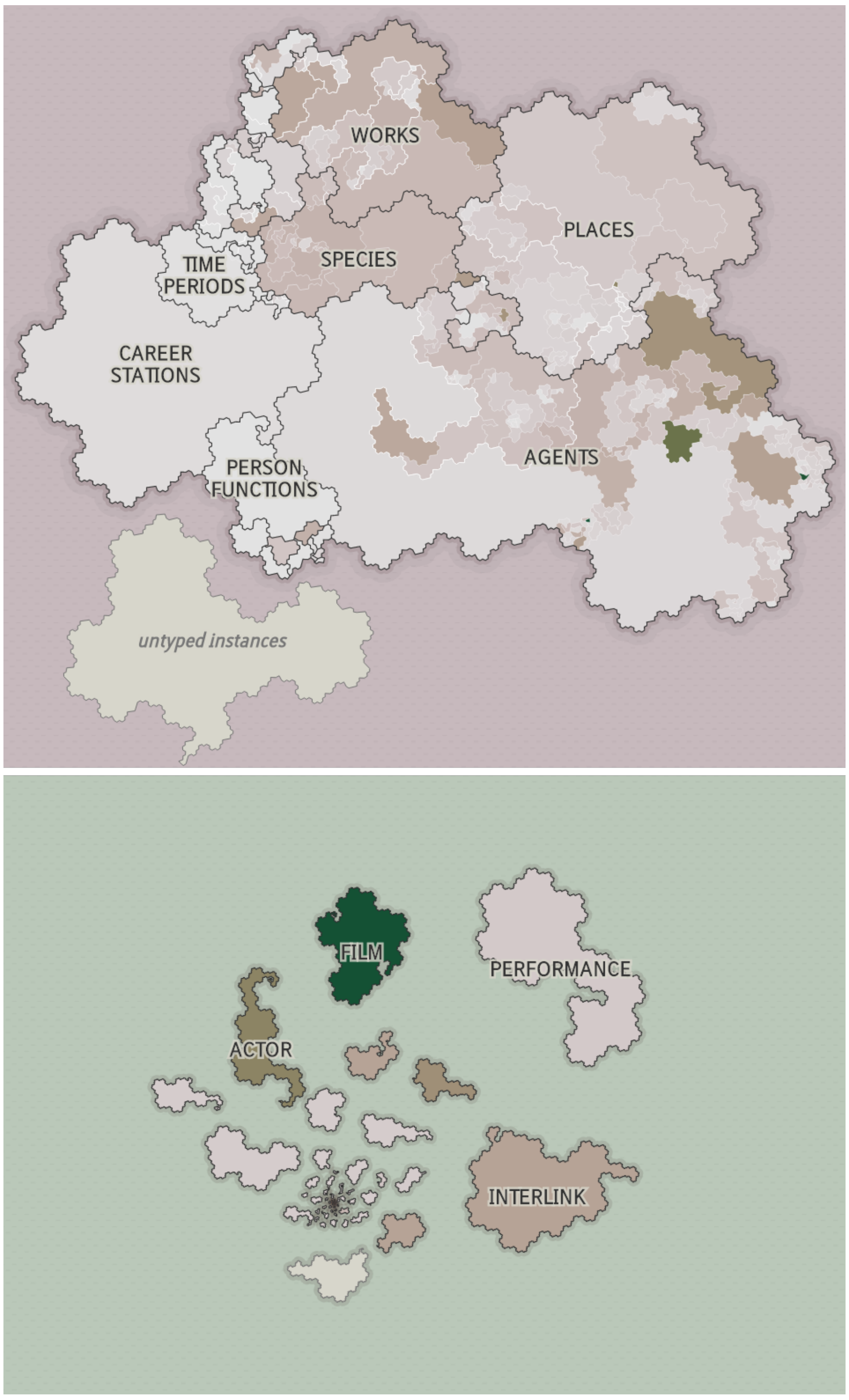


Figure 7.2: *Thematic maps of DBpedia and LinkedMDB, showing the density of object properties of each class (the darker, the more dense). By inspecting the maps, it can be seen that the most dense classes are Soccer manager, Horse trainer and Jockey for DBpedia and Film for LinkedMDB. The maps can also be compared in size since they share the same scale.*

7.2 NCBI Taxonomy database

The NCBI Taxonomy database[2] is the standard nomenclature and classification repository for the International Nucleotide Sequence Database Collaboration (INSDC). It includes the names and the taxonomic lineages of all known organisms. The classification is a phylogenetic taxonomy in which the structure corresponds to the evolutionary history of the tree of life. This particular structural organization contrasts the traditional approach defining the taxonomy with species differing from each other by a diagnosis stating their distinctive character. The top level of the taxonomy is composed of the following taxa: *Cellular Organisms*, *Viruses*, *Viroids*, *Unclassified sequences* and *Other sequences*.

Even if the total amount of entries the taxonomy contains is 1.482.059[3], it currently represents only 10% of the species on the planet.

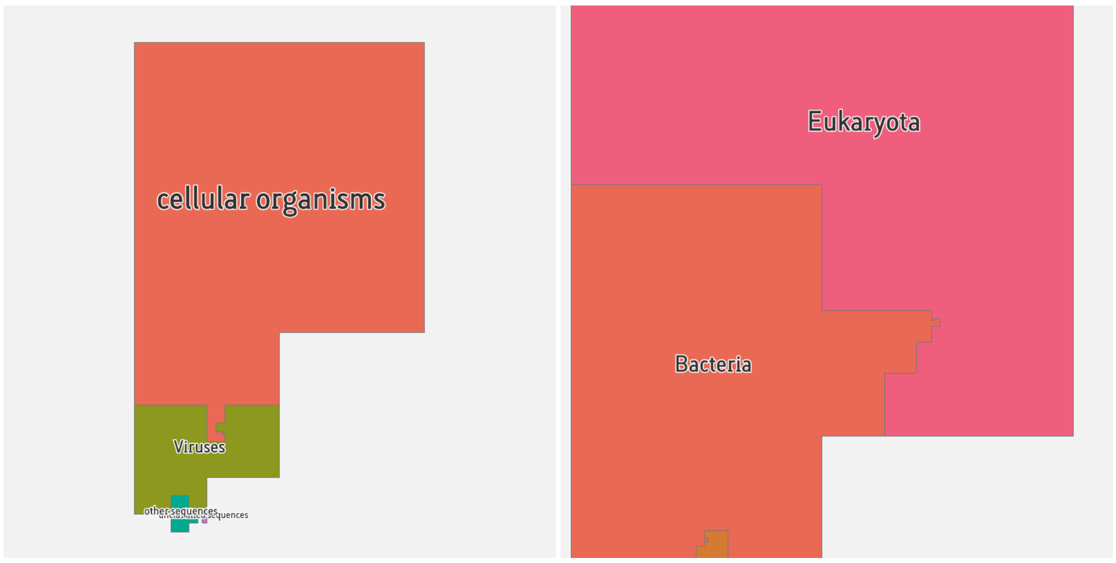


Figure 7.3: *On the left the initial overview of the visualization shows which are the main classes in the taxonomy (e.g., Cellular organisms and Viruses).* Cellular organisms *is the biggest class and by zooming in, it is exploded and replaced by its children* Eukaryota, Bacteria *and* Archaea *(on the right). An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/61187c0df9096d33820ac65d0467822c`.

Even if the NCBI taxonomy database is not modelled and published according to the Linked Data principles, it is possible to apply our approach and build an interactive application as shown in Figure 7.3 and 7.4. In order to implement it, a variation of the visualization pipeline, presented in Chapter 2, has been performed. In fact, a compound network was not generated by the data transformation process since it is a specific abstraction suitable for Linked Data but not so fitting for the NCBI data. More precisely, a variation of the compound network constituted only by a tree without a graph has been constructed.

The spatialization step has been performed using the curve of Hilbert and a squared

[2]http://www.ncbi.nlm.nih.gov/taxonomy
[3]Data was downloaded in July 2016

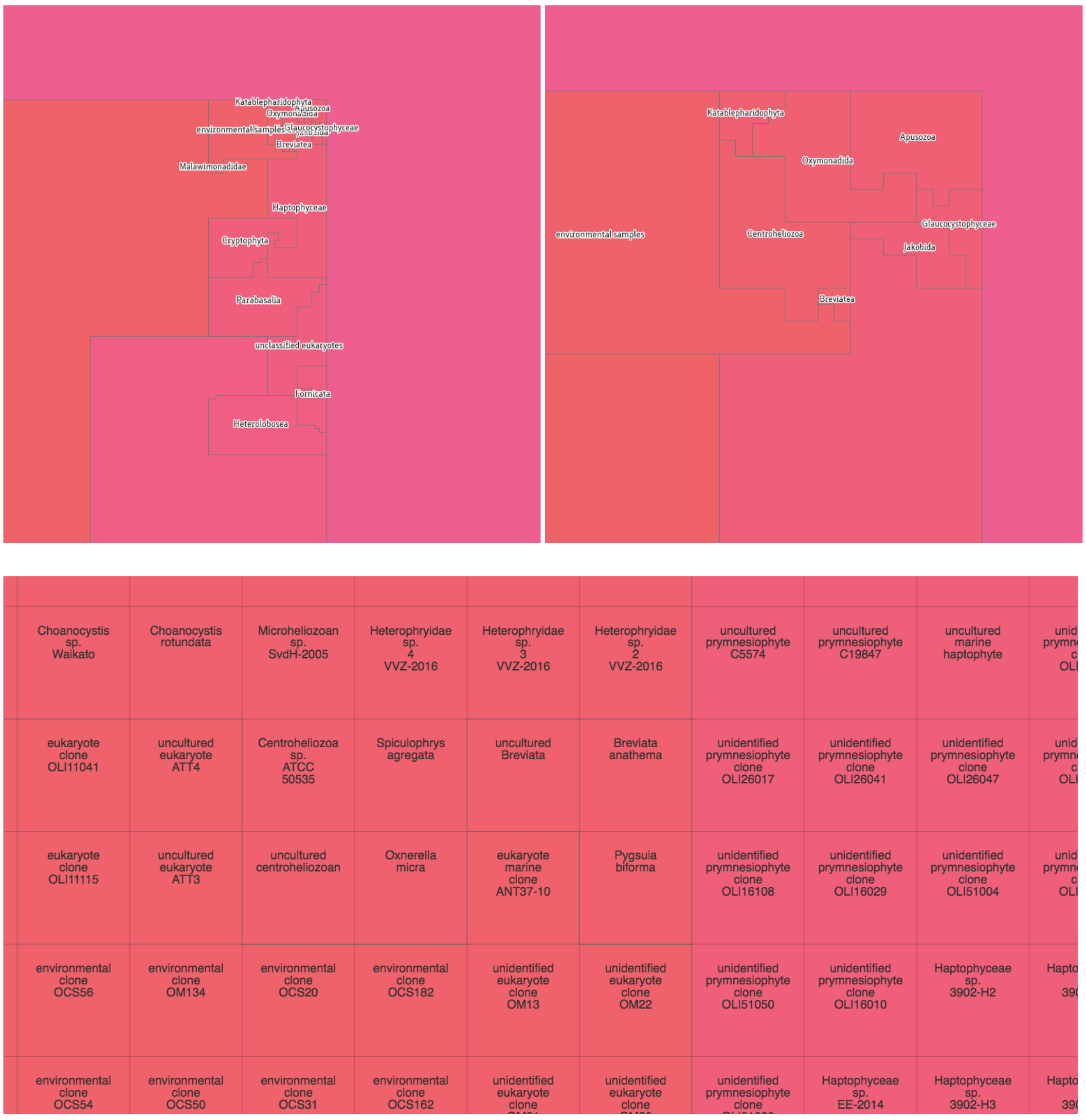

Katablepharidophyta
Apusozoa
Oxymonadida
environmental samples
Glaucocystophyceae
Jakobida
Breviatea
Malawimonadidae
Haptophyceae
Cryptophyta
Parabasalia
unclassified eukaryotes
Fornicata
Heterolobosea
Katablepharidophyta
Apusozoa
Oxymonadida
environmental samples
Centroheliozoa
Glaucocystophyceae
Jakobida
Breviatea
Choanocystis sp. Waikato
Choanocystis rotundata
Microheliozoan sp. SvdH-2005
Heterophryidae sp. 4 VVZ-2016
Heterophryidae sp. 3 VVZ-2016
Heterophryidae sp. 2 VVZ-2016
uncultured prymnesiophyte C5574
uncultured prymnesiophyte C19847
uncultured marine haptophyte
eukaryote clone OLI11041
uncultured eukaryote ATT4
Centroheliozoa sp. ATCC 50535
Spiculophrys agregata
uncultured Breviata
Breviata anathema
unidentified prymnesiophyte clone OLI26017
unidentified prymnesiophyte clone OLI26041
unidentified prymnesiophyte clone OLI26047
eukaryote clone OLI11115
uncultured eukaryote ATT3
uncultured centroheliozoan
Oxnerella micra
eukaryote marine clone ANT37-10
Pygsuia biforma
unidentified prymnesiophyte clone OLI16108
unidentified prymnesiophyte clone OLI16029
unidentified prymnesiophyte clone OLI51004
environmental clone OCS56
environmental clone OM134
environmental clone OCS20
environmental clone OCS182
unidentified eukaryote clone OM13
unidentified eukaryote clone OM22
unidentified prymnesiophyte clone OLI51050
unidentified prymnesiophyte clone OLI16010
Haptophyceae sp. 3902-H2
environmental clone OCS54
environmental clone OCS50
environmental clone OCS31
environmental clone OCS162
unidentified eukaryote clone
unidentified eukaryote clone
unidentified prymnesiophyte clone
Haptophyceae sp. EE-2014
Haptophyceae sp. 3902-H3

Figure 7.4: *Intermediate levels of zoom show intermediate classes in the hierarchy. By additionally zooming into the map it is possible to reach the maximum level of detail showing single tiles containing the labels of their corresponding species.*

tessellation. On the left-side of Figure 7.3 the overview of the data set is depicted and shows that *Cellular organisms* and *Viruses* are the main classes of the taxonomy in terms of size. By zooming in, the regions of the maps are exploded and replaced by their sub-regions (Right side of Figure 7.3 and on the top of Figure 7.4). As shown on the bottom of Figure 7.4 the maximum level of detail shows the squared tiles corresponding the species of the taxonomy.

7.3 Similarity-based resource placement

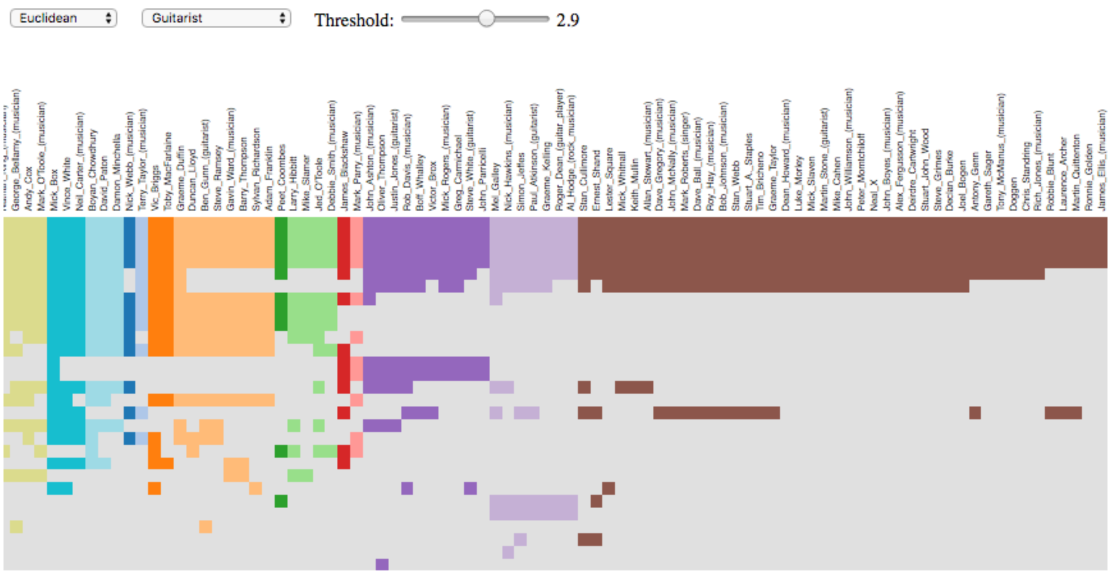
In this section, an experimental work, regarding the similarity between the elements of a data set and their positioning, is presented. So far, the arrangement of the elements of a given data set has been always described as the result of a spatialization process based on the use of a space-filling curve. This procedure guarantees that elements belonging to the same class are adjacently placed forming compact regions. However, no specific positioning has been established within the regions. In other words, by looking at the elements inside a region there is no criteria that keep close two elements except the fact that they belong to the same class. What if the elements were placed within regions according to a certain criteria based on their similarity?

The following experiments have been conducted using the data of DBpedia since it presents a strong imbalance between the amount of instances and classes. This peculiar feature produces regions, at the bottom of the regions hierarchy, composed of thousands of instances that could be hardly explored since no more hierarchical distinctions are available at this level. Therefore, it is possible to extend the hierarchy of DBpedia by applying hierarchical clustering algorithms for additionally grouping instances.

The first experiment, shown in Figure 7.5, tries to cluster the instances of 6 randomly selected classes (i.e., Guitarist, Poet, Cheese, Animanga Character, Monument and Pope). The instances are placed on the columns while their predicates on the rows forming a matrix. The cells of the matrix show whether an instance holds a certain predicate (coloured cell) or not (empty cell). Therefore, each column describes an instance in term of their predicates that in turn constitute a vector composed by zeros and ones. Hierarchical clustering can be then executed by giving as input:

- The vectors describing the instances;
- A *distance metric* for measuring the distance between the two items expressed as vectors. Three metrics have been chosen: the *Euclidean distance*, the *Manhattan distance* and the *Max distance*;
- A *type of linkage* for determining the distance between two clusters. The available kind of linkage are Average, Single and Complete;
- A *stopping criterion* also called threshold. When every cluster is more than threshold distance apart, clustering is stopped and the current set of hierarchies is returned.

The results of the clustering are then used in order to sort the instances and keep the most similar ones adjacent in the sequence they are depicted in the matrix. Using this ordering along with a coloured scale it is possible to see the clusters directly within the matrix. The predicates on the rows have been sorted in descending order from the most



Euclidean

Guitarist

Threshold: 2.9

Figure 7.5: *A zoomed portion of the matrix showing the results of the hierarchical clustering algorithm performed on the elements of some classes of DBpedia. Instances are placed on the columns while predicates on the rows. Each column is a vector describing an instance in term of predicates. Similar instances are kept close and depending on the threshold coloured according to the results of the clustering process. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/cc44f1a671e3ce135775`.

frequent to the least frequent one. Some insights can be seen exploring the visualization, for instance by selecting the *Pope* class, it is possible to see that different kinds of popes are described with similar sets of predicates. In fact, by setting the threshold of the hierarchical clustering to 2.9 with the Euclidean distance, the popes of Alexandria, the antipopes and the popes named John are grouped in different clusters. The class of *Cheese* is characterized by a very low amount of predicates and its instances are easily grouped since they are very similar in terms of the predicates they hold.

The second experiment, shown in Figure 7.6, has been adopted, starting from the previous one, for depicting clusters on the maps instead of using matrices.

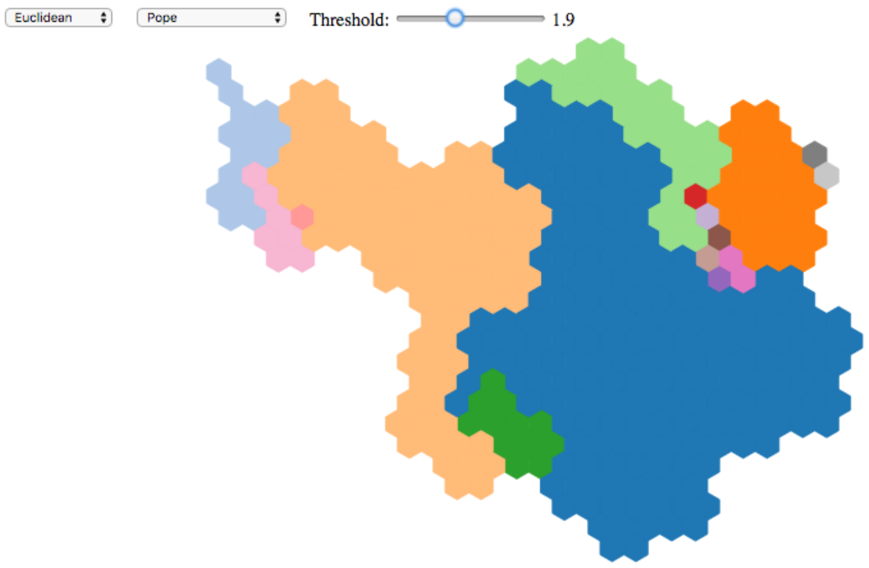


Figure 7.6: *The evolution of the previous experiment uses a map instead of a matrix. Clusters are shown as the regions of a certain class that can be selected using the dropdown menu on the top. Every time the value of the threshold is changed, the hierarchical clustering is executed and regions are updated on the map. An interactive version of this visualization can be consulted at* `http://bl.ocks.org/fabiovalse/f290de8239df9e5ee1d8`.

CHAPTER 8

Conclusion

THE work presented in this thesis consists of an approach for visualizing an entire Linked Data set. Indeed, its peculiarity is the overall visualization that comprises all the information composing an RDF graph. The ontology, its classes and the relative instances are all represented in different forms within a visualization based on cartographic principles. The visualizations resemble traditional geographical maps even if abstract data are arranged on the space instead of geographic places. In order to generate these maps, a spatialization process that assigns a shape, a size and a position on a two-dimensional space is performed. In particular, a procedure based on the use of space-filling curves has been adopted and a novel technique, exploiting the fractal nature of these curves, has been devised for spatializing data in a scalable and efficient way as demonstrated by the prototype presented in Chapter 6.

The interactive mechanisms characterizing these visualizations provide an intuitive method for the exploration of the maps as illustrated in Chapter 5 and 6. The cartographic metaphor allows to reuse the map-reading abilities, previously learned by humans, to get insights from complex and structured sources of data such as Linked Data.

In order to test its efficacy, the approach has been applied to the well-known Linked Data sets of DBpedia and LinkedMDB by developing two distinct Web applications as previously described in Chapter 6. By examining the outcome of the evaluation, we find both strengths and weaknesses, which overall characterize a promising approach.

- The overview initially presented when the map-like visualization is loaded is very effective in communicating which are the main ontological classes of a data set. In fact, the area encoding technique used by our visualization supports users in getting this information.

- The lines representing the relationships between the instances turned out to be very powerful. In fact, once a target instance is selected, the visualization immediately gives a gist of the classes of the other instances to which the target is related by drawing red arcs over the map for expressing the relationships. This feature is extremely useful when two or more sets of lines, characterizing distinct instances, are compared, since they can provide additional insights and patterns. Unfortunately, there is no feature yet allowing to simultaneously comparing sets of lines. A first improvement would be to support two sets of lines and color them with different colors to let users clearly distinguish the two sets.
- Thematic maps represent a valid instrument for showing even more information about the data set. They can be used as additional informative layers providing for example statistical estimations about RDF triples or more features about the ontology.
- The zoom mechanism for filtering the data set allows users to concentrate their attention only on some specific parts in a very intuitive way as the modern online maps services provide. An advancement could be the inclusion of a mini overview window for allowing users to orient themselves once the map is zoomed in.
- The zoom is the main instrument enabling the navigation of the hierarchical structure of the ontology and it actually gives good results but only for few levels. The more the map is zoomed the more it becomes difficult to precisely perceive region containment since the map actually uses only the stroke width for characterizing the depth levels of the hierarchy. More methods for differentiating the hierarchical levels, defined by the boundaries of the regions, are needed in order to make the hierarchy fully comprehensible. For instance, colours, stroke width, line styles could be combined together in order to make the hierarchy clearer.
- The label placement is one of the main challenges faced by map producers. In our approach, this feature has been simply implemented by placing the labels of the classes in the centroid of every region. This method produces maps in which labels can overlap each other or could be placed on a point outside the boundaries of their corresponding region. While the former case is caused by the excessive length of the labels, the latter is due to the particular shape characterizing concave regions. More sophisticated techniques could be introduced for producing a better layout that would make the exploration of the maps even clearer. For instance, an advanced version of the current semantic zoom could compute the proper level of zoom at which each label should be shown, by analyzing its length and the available space its corresponding region owns. Instead of horizontally placing all the labels, some of them could be slightly rotated to fit inside the boundaries and to better cover the space of their corresponding region.

A relevant discussion regards the requirements data have to respect in order to be compliant with the approach.

- The hierarchical structure describing ontologies is an advantageous characteristic to make use of, since it provides an organized layout to the visualizations. However, it defines a precise condition that must be respected by the input data. In fact, there are particular situations in which ontologies are modelled as graph structures and the approach would result to be not applicable. In this case, it is worth transforming the graph only if a reasonably small amount of links have to be removed in order to obtain a hierarchy.

- An instance node can be connected to multiple class nodes only if they are compatible as described in Chapter 5. Thus, data sets presenting incompatible class nodes result to be not compliant with our approach. A solution to this compatibility problem would be the duplication of the instance nodes for placing them in different regions within the map. Even if this solution seems to be inadequate, it has been already adopted by some Linked Data sets in their data modelling phase. For example, LinkedMDB creates different instance nodes for those people having multiple professions, such as actors that are also directors and film producers.

Among the performed experimental works, it is worth to mention the studies carried out on similarity. The intent of this analysis is to decrease the high imbalance between the number of classes and the amount of instances typically characterizing Linked Data sets. The problem clearly emerges during the user exploration. In fact, once a high level of zoom is reached, hundreds and sometimes even thousands of instances are displayed without a precise arrangement in the space except the fact that they belong to the same ontological class. Therefore, a similarity criterion has been devised in order to keep similar instances close. In this way it is possible to extend the hierarchy of the ontology with new “classes" generated by the clustering of the instances. These additional levels could be included in the hierarchy and then would visually produce a greater amount of nested regions within the visualization that would help users in their exploration.

By following the city metaphor, a distinctive instance of a region is represented as its capital. This is not an attempt at making maps more similar to the traditional geographic ones, but a way for making more explicit what a certain region contains. Since it is a experimental feature, the choice of the cities has been manually performed. Thus, it would be interesting to devise a method that automatically chooses cities by exploiting some characteristics present in the data. For instance, the most connected instance in term of relationships defined by the RDF triples could be used as the representative city of a region.

Regarding the level of flexibility, the approach provides (as described in Section 5.5) the possibility of setting different configurations depending on the specific case, and allows to move the computation offline and online according to the necessity. Moreover, even if the approach has been mainly presented for solving the problems characterizing the Semantic Web, as illustrated in Chapter 6, it can be applied to non-Linked Data sources as well.

Furthermore, the map can be enriched with other features such as the thematic maps discussed in Chapter 7. It can be seen as a base visualization on top of which other functionality can be implemented. For example, the results of the execution of SPARQL queries could be displayed by highlighting the resulting instances on the map. A feature similar to the one proposed by RelFinder (Chapter 3) could be included for showing which are the paths between two instances.

8.1 Closing remarks

The main goal of the thesis was to provide an approach for visualizing a data set in its entirety and enabling users to learn and understand how it is composed. Considering this intention, the approach discussed in the thesis surely represents a possible solution to the problem since, as demonstrated by the prototypes developed, the combination of the cartographic approach with a zoom behaviour permits to visualize large amounts of complex data in an interactive and natural way. Moreover, the choice of structuring the process of exploration according to the Shneiderman Mantra additionally contribute to this aim. The general structure of a data set is shown as first view with a map, then a zoom mechanism allows to deeper explore the parts on which a user is interested in, finally details about the tiniest resources of the map can be requested.

The approach is promising considering the prototypes developed and the results of the evaluations performed. However, more case studies are necessary in order to further evaluate the approach on different scenarios and additionally improve it. In conclusion, the work performed is answering the initial goal of the thesis and it represents a nucleus of ideas, techniques and experiments that can be used as a starting point for further investigating the possibility of entirely representing sets of data.

Bibliography

[1] Jans Aasman and Ken Cheetham. Rdf browser for data discovery and visual query building. In *Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2011), Co-located with ACM IUI*, page 53, 2011.

[2] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[3] Matteo Abrate. *Data Cartography: atlases and maps for non-geographical data*. PhD thesis, University of Pisa, 2014.

[4] Daniel Archambault, Tamara Munzner, and David Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *Visualization and Computer Graphics, IEEE Transactions on*, 2008.

[5] David Auber, Daniel Archambault, Romain Bourqui, Antoine Lambert, Morgan Mathiaut, Patrick Mary, Maylis Delest, Jonathan Dubois, and Guy Mélançon. *The tulip 3 framework: A scalable software library for information visualization applications based on relational data*. PhD thesis, INRIA, 2012.

[6] David Auber, Charles Huet, Andrew Lambert, Benjamin Renoust, Arnaud Sallaberry, and Agnes Saulnier. Gospermap: Using a gosper curve for laying out hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 19(11):1820–1832, 2013.

[7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[8] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.

[9] Sean Bechhofer and Alistair Miles. Skos simple knowledge organization system reference. *W3C recommendation, W3C*, 2009.

[10] Christian Becker and Chris Bizer. Marbles linked data engine, 2009.

[11] Tim Berners-Lee. Design issues: Linked data, 2006.

[12] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd international semantic web user interaction workshop*, volume 2006, page 159. Athens, Georgia, 2006.

[13] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[14] Robert P Biuk-Aghai, Muye Yang, Patrick Cheong-Iao Pang, Wai Hou Ao, Simon Fong, and Yain-Whar Si. A map-like visualisation method based on liquid modelling. *Journal of Visual Languages & Computing*, 31:87–103, 2015.

[15] Chris Bizer and Tobias Gauß. Disco - hyperdata browser, 2007.

[16] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.

[17] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[18] Maged N Kamel Boulos. The use of interactive graphical maps for browsing medical/health internet information resources. *International Journal of Health Geographics*, 2(1):1, 2003.

[19] Dan Brickley and Ramanathan V Guha. {RDF vocabulary description language 1.0: RDF schema}. 2004.

[20] Willard Cope Brinton. *Graphic methods for presenting facts*. Engineering magazine company, 1917.

[21] Willard Cope Brinton. *Graphic presentation*. Brinton associates, 1939.

[22] Josep Maria Brunetti, Sören Auer, and Roberto García. The linked data visualization model. In *International Semantic Web Conference (Posters & Demos)*, 2012.

[23] Josep Maria Brunetti, Sören Auer, Roberto García, Jakub Klímek, and Martin Nečaskỳ. Formal linked data visualization model. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, page 309. ACM, 2013.

[24] Diego Valerio Camarda. Lod view, 2014.

[25] Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. Lodlive, exploring the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 197–200. ACM, 2012.

[26] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[27] Matthew Chalmers. Using a landscape metaphor to represent a corpus of documents. In *European Conference on Spatial Information Theory*, pages 377–390. Springer, 1993.

[28] Matthew Chalmers and Paul Chitson. Bead: Explorations in information visualization. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 330–337. ACM, 1992.

[29] Ed Huai-hsin Chi and John T Riedl. An operator interaction framework for visualization systems. In *Information Visualization, 1998. Proceedings. IEEE Symposium on*, pages 63–70. IEEE, 1998.

[30] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[31] Aba-Sah Dadzie, Matthew Rowe, and Daniela Petrelli. Hide the stack: toward usable linked data. In *The Semantic Web: Research and Applications*, pages 93–107. Springer, 2011.

[32] Mark de Berg, Krzysztof Onak, and Anastasios Sidiropoulos. Fat polygonal partitions with applications to visualization and embeddings. *arXiv preprint arXiv:1009.1866*, 2010.

[33] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, and L Andrea Stein. Owl web ontology language reference. *W3C Recommendation February*, 10, 2004.

[34] Marek Dudáš, Vojtěch Svátek, and Jindřich Mynarz. Dataset summary visualization with lodsight. In *The Semantic Web: ESWC 2015 Satellite Events*, pages 36–40. Springer, 2015.

[35] Marek Dudás and Vojtech Svátek. Discovering issues in datasets using lodsight visual summaries. In *VOILA@ISWC*, volume 1456 of *CEUR Workshop Proceedings*, page 77. CEUR-WS.org, 2015.

[36] Ivan Ermilov, Michael Martin, Jens Lehmann, and Sören Auer. Linked open data statistics: Collection and exploitation. In *Knowledge Engineering and the Semantic Web*, pages 242–249. Springer, 2013.

[37] Sara Irina Fabrikant and André Skupin. Cognitively plausible information visualization. 2005.

[38] Fernando Florenzano, Denis Parra, Juan L Reutter, and Freddie Venegas. A visual aide for understanding endpoint data.

[39] Emden R Gansner, Yifan Hu, and Stephen Kobourov. Gmap: Visualizing graphs and clusters as maps. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pages 201–208. IEEE, 2010.

[40] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[41] Martin Gronemann and Michael Jünger. Drawing clustered graphs as topographic maps. In *International Symposium on Graph Drawing*, pages 426–438. Springer, 2012.

[42] Florian Haag, Steffen Lohmann, Stephan Siek, and Thomas Ertl. Queryvowl: Visual composition of sparql queries. In *The Semantic Web: ESWC 2015 Satellite Events*, pages 62–66. Springer, 2015.

[43] Andreas Harth. Visinav: A system for visual search and navigation on web data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):348–354, 2010.

[44] Oktie Hassanzadeh and Mariano P Consens. Linked movie data base. In *Linked Data On the Web (LDOW)*, 2009.

[45] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[46] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. Relfinder: Revealing relationships in rdf knowledge bases. In *Semantic Multimedia*, pages 182–187. Springer, 2009.

[47] Philipp Heim, Jürgen Ziegler, and Steffen Lohmann. gfacet: A browser for the web of data. In *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW'08)*, volume 417, pages 49–58. Citeseer, 2008.

[48] Ivan Herman, Guy Melançon, and M Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6(1):24–43, 2000.

[49] Patrick Hoefler, Michael Granitzer, Vedran Sabol, and Stefanie Lindstaedt. Linked data query wizard: A tabular interface for the semantic web. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 173–177. Springer, 2013.

[50] Patrick Hoefler, Michael Granitzer, Eduardo E Veas, and Christin Seifert. Linked data query wizard: A novel interface for accessing sparql endpoints. In *LDOW*, 2014.

[51] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741–748, 2006.

[52] Yifan Hu. A gallery of large graphs. `http://yifanhu.net/GALLERY/GRAPHS/`, 2014.

[53] Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on*, pages 284–291. IEEE, 1991.

[54] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. Rql: a declarative query language for rdf. In *Proceedings of the 11th international conference on World Wide Web*, pages 592–603. ACM, 2002.

[55] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods—a survey. *ACM Computing Surveys (CSUR)*, 39(4):10, 2007.

[56] Daniel A Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on visualization and computer graphics*, 6(1):59–78, 2000.

[57] Jakub Klímek, Jiří Helmich, and Martin Nečaskỳ. Payola: Collaborative linked data analysis and visualization framework. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 147–151. Springer, 2013.

[58] Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[59] Donald Ervin Knuth. *The Art of Computer Programming: Fundamental Algorithms*. Addison-Wesley, 1968.

[60] H Koch. On a continuous curve without tangent constructable from elementary geometry. editor: Edgar, g.(1993). *Classics on fractals*, 1904.

[61] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[62] Joseph B Kruskal and James M Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[63] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[64] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[65] Denis Lukovnikov, Dimitris Kontokostas, Claus Stadler, Sebastian Hellmann, and Jens Lehmann. Dbpedia viewer-an integrative interface for dbpedia leveraging the dbpedia service eco system. In *LDOW*. Citeseer, 2014.

[66] Denis Lukovnikov, Claus Stadler, and Jens Lehmann. Ld viewer-linked data presentation framework. In *Proceedings of the 10th International Conference on Semantic Systems*, pages 124–131. ACM, 2014.

[67] Benoit B Mandelbrot. *The fractal geometry of nature*, volume 173. Macmillan, 1983.

[68] Nicolas Marie and Fabien Gandon. Survey of linked data based exploration systems. In *Proceedings of the 3rd International Conference on Intelligent Exploration of Semantic Data-Volume 1279*, pages 66–77. CEUR-WS. org, 2014.

[69] Michael Martin, Konrad Abicht, Claus Stadler, Axel-Cyrille Ngonga Ngomo, Tommaso Soru, and Sören Auer. Cubeviz: Exploration and visualization of statistical linked data. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 219–222. International World Wide Web Conferences Steering Committee, 2015.

[70] Daisuke Mashima, Stephen Kobourov, and Yifan Hu. Visualizing dynamic data with maps. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1424–1437, 2012.

[71] Suvodeep Mazumdar, Daniela Petrelli, and Fabio Ciravegna. Exploring user and system requirements of linked data visualization through a visual dashboard approach. *Semantic Web*, 5(3):203–220, 2014.

[72] Isabel Meirelles. *Design for information: an introduction to the histories, theories, and best practices behind effective information visualizations*. Rockport publishers, 2013.

[73] David Modjeska. Navigation in electronic worlds: A research review. *Toronto, Computer Systems Research Group, University of Toronto*, 1997.

[74] Tamara Munzner. *Visualization Analysis and Design*. CRC Press, 2014.

[75] Alberto Musetti, Andrea Giovanni Nuzzolese, Francesco Draicchio, Valentina Presutti, Eva Blomqvist, Aldo Gangemi, and Paolo Ciancarini. Aemoo: Exploratory search based on knowledge patterns over the semantic web. *Semantic Web Challenge*, 136, 2012.

[76] S Negru, S Lohmann, F Haag, et al. Vowl: Visual notation for owl ontologies, 2014.

[77] OpenLink-Software. Openlink data explorer, 2007.

[78] Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In *Internatioan Semantic Web Conference*. 2013.

[79] Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[80] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1419–1428. International World Wide Web Conferences Steering Committee, 2016.

[81] Oscar Peña, Unai Aguilera, and Diego López-de Ipina. Linked open data visualization revisited: a survey. *Submitted to Semantic. Web J*, 2015.

[82] Eric Prud'Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[83] Edward M Reingold and John S Tilford. Tidier drawings of trees. *Software Engineering, IEEE Transactions on*, (2):223–228, 1981.

[84] Hans Sagan. *Space-filling curves*. Springer-Verlag, 1994.

[85] Kai Schlegel, Thomas Weißgerber, Florian Stegmaier, Christin Seifert, Michael Granitzer, and Harald Kosch. Balloon synopsis: a modern node-centric rdf viewer and browser for the web. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 249–253. Springer, 2014.

[86] m Schraefel and David Karger. The pathetic fallacy of rdf. In *International workshop on the semantic web and user interaction (SWUI)*, volume 2006, 2006.

[87] Hans-Jorg Schulz. Treevis. net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011.

[88] Steven Schwartzman. *The words of mathematics: An etymological dictionary of mathematical terms used in English*. MAA, 1994.

[89] Juan Sequeda. Introduction to: Triplestores, 2013.

[90] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92–99, 1992.

[91] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996.

[92] Michael Sintek and Stefan Decker. Triple-an rdf query, inference, and transformation language. In *INAP*, pages 47–56, 2001.

[93] André Skupin. From metaphor to method: Cartographic perspectives on information visualization. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 91–97. IEEE, 2000.

[94] André Skupin. The world of geography: Visualizing a knowledge domain with cartographic means. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5274–5278, 2004.

[95] André Skupin and Sara Irina Fabrikant. Spatialization methods: a cartographic research agenda for non-geographic information visualization. *Cartography and Geographic Information Science*, 30(2):99–119, 2003.

[96] André Skupin and Sara Irina Fabrikant. Spatialization. 2007.

[97] John Stasko and Eugene Zhang. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 57–65. IEEE, 2000.

[98] Christos Stergiou and Dimitrios Siganos. Neural networks, 1996.

[99] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[100] R Core Team et al. R: A language and environment for statistical computing. 2013.

[101] Jason Vallet, Guy Melançon, and Bruno Pinaud. Jasper: Just a new space-filling and pixel-oriented layout for large graph overview. *Electronic Imaging*, 2016(1):1–10, 2016.

[102] Fabio Valsecchi. Spacetime: a two dimensions search and visualisation engine based on linked data. Master's thesis, University of Trento, 2013.

[103] Fabio Valsecchi, Matteo Abrate, Clara Bacciu, Maurizio Tesconi, and Andrea Marchetti. Dbpedia atlas: Mapping the uncharted lands of linked data. In *World Wide Web Conference, Linked Data on the Web*, 2015.

[104] Fabio Valsecchi, Matteo Abrate, Clara Bacciu, Maurizio Tesconi, and Andrea Marchetti. Linked data maps: Providing a visual entry point for the exploration of datasets. 2015.

[105] Fabio Valsecchi and Marco Ronchetti. Spacetime: a two dimensions search and visualisation engine based on linked data. In *Conference on Advances in Semantic Processing (SEMAPRO)*, 2014.

[106] Jeffrey Ventrella. *Brainfilling Curves-A Fractal Bestiary*. 2012.

[107] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[108] Marc Wick and C Boutreux. Geonames. *GeoNames Geographical Database*, 2011.

[109] Pak C Wong and Jim Thomas. Visual analytics. *IEEE Computer Graphics and Applications, 24 (5): 20-21*, 24(PNNL-SA-41935), 2004.

[110] Pak Chung Wong, Harlan Foote, Patrick Mackey, George Chin, Zhenyu Huang, and Jim Thomas. A space-filling visualization technique for multivariate small-world graphs. *IEEE transactions on visualization and computer graphics*, 18(5):797–809, 2012.

[111] Robert Alan Wright, Bruce Richmond, Andrew Odlyzko, and Brendan D McKay. Constant time generation of free trees. *SIAM Journal on Computing*, 15(2):540–548, 1986.

[112] Muye Yang and Robert P Biuk-Aghai. Enhanced hexagon-tiling algorithm for map-like information visualisation. In *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction*, pages 137–142. ACM, 2015.