An Interactive 3D Visualization for the LOD Cloud

Maria-Evangelia Papadaki, Panagiotis Papadakos, Michalis Mountantonakis, Yannis Tzitzikas
Institute of Computer Science, FORTH-ICS, GREECE, and
Computer Science Department, University of Crete, GREECE
{marpap|papadako|mountant|tzitzik}@ics.forth.gr

ABSTRACT

The LOD (Linked Open Data) cloud currently contains thousands of published datasets. Existing visualizations, like the Linking Open Data cloud diagram, are useful for getting an overview of its size, the datasets and their connectivity. An interesting question is whether we could come up with more informative and more interactive visualizations that could make evident more features of the datasets for aiding the inspection and the discovery of related datasets. To this end we propose an interactive 3D visualization that adopts the metaphor of *urban area*. In brief, each dataset is visualized as a building, whose features (e.g. volume) reflect various dataset's features (e.g. number of triples), while the proximity of the buildings (and other features) indicates the commonalities of the datasets. The introduced approach supports various shapes of buildings and various placement algorithms: mountainside, orthogonal spiral, concentric spiral, and similarity-based adaptations of force-directed algorithms. The visualization is interactive, i.e. it allows the user to zoom in any part of the model, to change the perspective, to change the shape of the buildings and their placement, to see all the connections or only those of one dataset, and others. The paper details the construction process and provides examples over real datasets including the entire LOD cloud, and describes the pros and cons of each layout.

KEYWORDS

Linked Data, Connectivity of Linked datasets, Interactive 3D Visualization

1 INTRODUCTION

During the last years we observe an increasing trend towards publishing data as LOD. Thousands of datasets have been published and various visualizations that give an overview of their number and interconnections have been proposed (e.g. see [1, 7, 8]). The classical visualization of the LOD cloud (Figure 1(left)), depicts each dataset as a circle (whose size indicates the size of the dataset in triples). The commonalities between two datasets (in terms of common URIs) are made evident by an edge that connects the dataset's circles. Such visualizations are useful for getting an overview of the entire LOD cloud, or for a part of it, or for a particular set of RDF datasets. There are various visualization-driven tasks. In our work we focus mainly on tasks related to *datasets inspection*, *datasets monitoring*, *dataset selection* and *navigation* across multiple linked datasets. The basic question we address here, is: *can we come up with visualizations of the LOD cloud which are more informative* (i.e. which can make evident more "features" of the datasets) and *are easily conceivable*? Based on this motivation, in this paper we propose an interactive 3D visualization that adopts a quite familiar metaphor, specifically that of an *urban area* where each dataset is visualized as a *building*. An indicative screenshot of the LOD cloud according to the interactive 3D visualization that we propose, is shown in Figure 1(right). In a nutshell the contributions of this paper are: (i) it introduces and motivates a novel interactive 3D model for LOD datasets that adopts the metaphor of urban area, (ii) it introduces several variations of the model, and discusses the pros and cons of each one, and (iii) it demonstrates the application of the model over the datasets of each domain (government, media, etc.) and the entire LOD cloud. The rest of this paper is organized as follows: §2 describes the context, §3 describes the main components of the interactive 3D model, and its application, §4 describes the implementation of the visualization system as well as directions that are worth further work and research, and finally §5 concludes the paper. A running prototype is already available to the community and it is accessible through **http://www.ics.forth.gr/isl/3DLod** (needs a recent web browser supporting WebGL).

2 CONTEXT

Visualization has been recognized as important for *dataset discovery* and *dataset selection* [10], which consist two of the most emerging challenges for the web of data [5, 9]. A number of visualization approaches and tools for Linked Data have been proposed, some indicative of which are described in [6]. The most widely known visualization diagram of the LOD is the 2D Linking Open Data cloud diagram, which consists of datasets that have been published in Linked Data format by contributors to the Linking Open Data community project and other individuals and organisations. It is based on metadata collected and curated by contributors to the *datahub.io* as well as on metadata extracted from periodic crawls of the Linked Data web. The 2014 crawled version of the diagram is shown in Figure 1(left). We refer to the Linking Open Data cloud that was available from 2014-08-30 to 2017-01-25[1] that contains datasets from the following nine domains (in parenthesis the percentages of datasets that fall in each category): government (23.85%), publications (23.33%), social web (15.78%), life sciences (11.05%), cross-domain (7.19%), user-generated content (7.36%), geographic (4.21%), media (3.68%), and linguistics (3.50%). The size of the circles corresponds to the number of triples in each dataset. Only *five* sizes of circles (very large, large, medium, small, very small) are supported each corresponding to a particular size interval (> 1 B, 10M-1B, 500K-10M, 10K-500K, < 10K resp.). The arrows between two circles indicate the existence of at least 50 links between the corresponding two datasets. A link is considered as an RDF triple where subject and object URIs are in the namespaces of different datasets, while the direction of the arrows indicates the dataset that contains the links. The thickness of the arrow corresponds to the number of links. Three levels of thickness are supported (thin, medium, thick) each corresponding to one interval $((0, 1K), [1K, 100K)$ and $[100K, \infty)$ respectively). Finally,

[1] Accessible through http://lod-cloud.net/versions/2014-08-30/lod-cloud_colored.svg

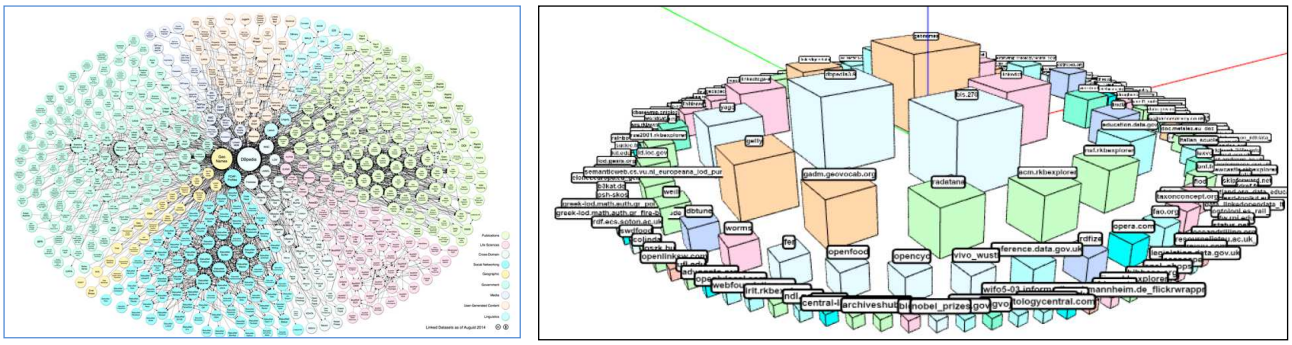

Figure 1: *Left*: The LOD cloud diagram. *Right*: One perspective of the introduced interactive LOD 3D model.

each circle is colored differently for indicating the 9 different domains of the datasets. A new version of the Linking Open Data cloud diagram was released on 2017-01-26.[2] That version contains almost double the number of datasets (i.e. 1163). Datasets are again visualized as circles however only three sizes of circles (large, medium, small) are supported. The links are interactive and their direction is indicated through color. However, the clustering of the datasets is not favorable in all cases and the labels are less readable in comparison to the 2014-2017 version of the diagram.

3 AN INTERACTIVE URBAN 3D VISUALIZATION FOR LOD DATASETS

3.1 Dataset Notations

Let $S = S_1, \dots S_k$ be the set of datasets. Each dataset S_i consists of a set of triples (i.e., a set of `subject-predicate-object` statements), denoted by $triples(S_i)$. We shall use U_i to denote the URIs, L_i to denote the literals and BN_i to denote the blank nodes that appear in $triples(S_i)$. Hereafter, we consider only those URIs that appear as *subjects* or *objects* in a triple as our primary focus is on the data (not on schema). The number of *common URIs* between two datasets S_i and S_j, is given by $|U_i \cap U_j|$. We define the *Links* between two datasets as follows: $Links_{i,j} = U_i \cap U_j$. If T is a set of triples, then we can define the *degree* of a URI e in T as: $deg_T(e) = |\{(s,p,o) \in T \mid s = e \text{ or } o = e\}|$, while for a set of URIs E we can define their average degree in T as $deg_T(E) = avg_{e \in E}(deg_T(e))$. Now for each dataset S_i we can compute the *average degree* of the elements in U_i by considering $triples(S_i)$, i.e.: $Deg(U_i) = avg_{e \in U_i}(deg_{triples(S_i)}(e))$.

3.2 Buildings Representation

The main idea is that we visualize each dataset S_i as a *building* b_i. The volume of each building represents the number of triples of the respective dataset ($|triples(S_i)|$). As regards the types of the buildings, we support the following options: (a) *cubes*, (b) *"context"-dependent cuboids*, and (c) *"feature"-based cuboids*.

In (a), each dataset S_i is represented by a *cube* with edge length equal to $\sqrt[3]{|triples(S_i)|}$.

In (b) we use *"context-dependent" cuboids*. The footprint of the buildings is computed based on either the biggest dataset (b_{Big} mode) or the smallest dataset (b_{Small} mode). In the b_{Big} mode the building of the biggest dataset is a cube, while in the b_{Small} mode the cube corresponds to the smallest dataset. Consequently, the buildings of the datasets that have enough triples tend to become skyscrapers.

In (c), i.e. *"feature-based" cuboids*, the shape depends on the features of the corresponding datasets. Since $|triples(S_i)| \approx (|U_i| + |L_i| + |BN_i|) * Deg(U_i)$, the height of the building is set to be analogous to $|U_i| + |L_i| + |BN_i|$, and the footprint of the building analogous to $Deg(U_i)$. Specifically, assuming square footprints, we have $height(b_i) = |U_i| + |L_i| + |BN_i|$ and $width(b_i) = \sqrt{Deg(U_i)}$. The volume of the building b_i approximates $|triples(S_i)|$; if its degree is low it will become a high building with a small footprint, whereas if its degree is high then the building will be less tall but will have a big footprint.

For getting building sizes that resemble those of a real urban area, a calibration is required. For this reason we introduce an additional parameter F, through which we can obtain the desired average ratio of height/width of the buildings. Specifically, let r be the desired ratio (e.g. 3 for three-floor buildings) provided by the user. We can add a parameter F to the definition of *height* and *width*: $height(b_i) = (|U_i| + |L_i| + |BN_i|)/F$ and $width(b_i) = \sqrt{Deg(U_i) * F}$. Note that any positive value of F yields a pair of $height(b_i)$ and $width(b_i)$ that preserves the volume. What is left to do is to select the F for obtaining the desired average ratio r. This reduces to finding the F such that $r = avg\ \{\ \frac{height(b_i)}{width(b_i)} \mid 1 \le i \le k\}$. The solution of this equation is: $F = \sqrt[3]{\left(\frac{\sum_{i=1}^{|S|}(|U_i|+|L_i|+|BN_i|)/\sqrt{Deg(U_i)}}{r*|S|}\right)^2}$. Obviously instead of *avg* one can specify the *min* or *max* desired ratio and in that case the formula is changed accordingly, e.g., for the max desired ratio, we should first compute for each dataset the following number: $F_i = \sqrt[3]{\left(\frac{(|U_i|+|L_i|+|BN_i|)/\sqrt{Deg(U_i)}}{r}\right)^2}$. Then, we should sort all F_i in descending order and select the max F_i. By selecting the max F_i as the F value in all $height(b_i)$ and $width(b_i)$, then that guarantees that all buildings will have ratio $\le r$.

3.3 Placement of the Buildings

Below we describe four different building layout approaches, that our system supports.

1. Mountainside Layout. The k buildings are placed in an orthogonal $\lceil\sqrt{k}\rceil \times \lceil\sqrt{k}\rceil$ grid. The biggest building is placed in one edge of the square area. The second bigger is placed next to the first, and so on, until reaching the end of a row, where it continues the same procedure in the next one until there are no

[2] by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak, http://lod-cloud.net/

Table 1: Comparison of visualizations for Linked Datasets

Aspect	Mountain Side	Orthogonal Spiral	Cyclic Spiral	Similarity-based layout	LOD Cloud Diagram (2D)
Accurate size	✓	✓	✓	✓	only 5 sizes
Features (e.g. Degree)	✓	✓	✓	✓	
Connectivity	✓	✓	✓	✓	✓
Interactive	rich	rich	rich	rich	poor (only 2D with zoom)
Time Complexity	$O(n)$	$O(n)$	$O(n)$	$O(e * n^2)$	not mentioned
Distinctive characteristic	Readability	Fast overview of datasets' sizes	Effective space exploitation even for power law distributed datasets	Focus on connectivity (less edge crossings)	Label readability, connections, clustered datasets by domain

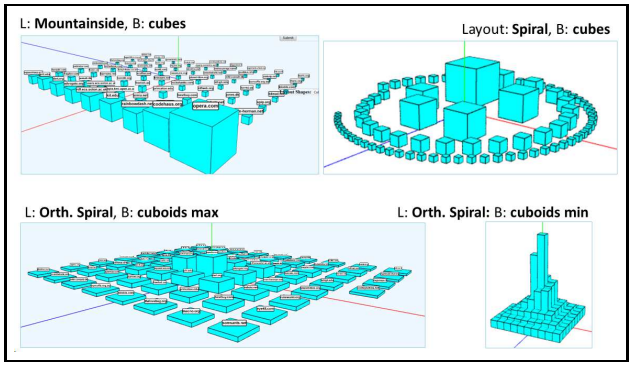

Figure 2: Visualizations of the *social* datasets in LOD 2014

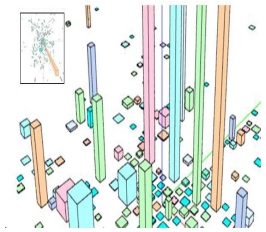

Figure 3: Similarity-based layout

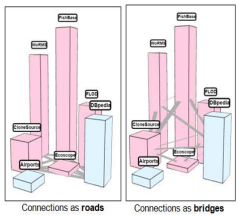

Figure 4: Two ways to visualize the connections

buildings to draw. The result resembles a *mountainside* (Figure 2 - upper left).

2. Orthogonal Spiral. The k buildings are placed in an orthogonal $\lceil\sqrt{k}\rceil \times \lceil\sqrt{k}\rceil$ grid (see the two screenshots at the bottom part of Figure 2). The biggest building is placed in the centre of the area (summit). The process continues by adding growing enclosing squares of size N. For example, the first contains 8 squares (3 above of the summit, one left and one right of the summit and 3 below the summit). The next enclosing square contains 16 more buildings and so on. Each building is drawn following the clockwise direction. The result resembles a mountain whose summit is at the centre of the 2D area. One shortcoming of this algorithm is that if we represent buildings with cubes then this algorithm yields very sparse peripheral areas. This was actually the motivation for the subsequent algorithm.

3. Cyclic Spiral. Based on the weaknesses of the previous layout algorithms, we identified the following requirements for a better layout algorithm:

(a) bigger buildings should be placed at the center
(b) a spiral-like placement seems beneficial as it would result to a round coverage of the space,
(c) collisions should never occur,
(d) no big empty spaces, especially in the outer area that hosts the majority of the buildings which are small.

For the above requirements, we devised a new 2D placement algorithm called *Concetric Spiral*. The buildings, in an descending order with respect to their size, are placed on concentric rings. The radius of the first (smallest) ring is the size of the biggest building. The placement of the subsequent buildings is done as follows. We compute the minimum chord that is required to avoid collisions based on the sizes of the current and the previous building. Then we compute the corresponding angle, and we place the new building at the corresponding spot of the circle. The sought angle is $\theta = 2\arcsin(\frac{chord}{2 \cdot radius})$. Just before we reach 2π, we start the next bigger ring whose radius, is the radius of the previous ring increased by the size of the last drawn building (plus a number accounting for "roads"). In this way the concentric rings become denser as the buildings get smaller avoiding the unnecessary empty spaces. The algorithm is appropriate for sets of datasets whose sizes vary a lot, even if they exhibit a power law distribution (i.e. very few big datasets and too many small ones, see [3, 9] for measurements about current RDF datasets). A screenshot of the layout based on *Cyclic Spiral* is shown in Figure 1 - right and Figure 2 - upper right. The algorithm has $O(n)$ time complexity (n is the number of buildings).

4. Similarity-based layout. According to this algorithm, the more commonalities two sources have (common URIs, common literals, `owl:sameAs` relationships, etc.) the closer the corresponding buildings are placed. One way to specify the location of each building is to adopt a *force-directed placement algorithm*. In our case, we have modified the Fruchterman-Reingold force directed algorithm [4] as adapted to `three.js`[3]. This algorithm satisfies the following two principles: a) vertices connected by an edge should be drawn near each other and b) vertices should not be drawn too close to each other. Figure 3 shows an indicative layout produced by the similarity-based algorithm.

Comparison. Table 1 summarizes the distinctive characteristics points of each visualization approach including the 2D LOD Cloud diagram. The value "rich" in the line "interactive" refers to interactive selection, zooming, panning, rotation, and control of visibility of labels and connections.

3.4 Visualizing the Links of Datasets

If there are links between two datasets S_i and S_j then a line segment is created, resembling a *road* that connects the corresponding buildings (see the left side of Figure 4). The links can be also visualized as *bridges* (see the right side of Figure 4). The width of these bridges/roads, indicates the *strength of the connection* that

[3]https://github.com/davidpiegza/Graph-Visualization

the correlated datasets have, and it is calculated by the division of the number of links between S_i and S_j with the number of links of the most connected pair (i.e., maxLinks): $width(i,j) = \frac{|Links(i,j)|}{|maxLinks|}$.

3.5 Application Cases

We downloaded manually 287 RDF datasets including their content (i.e., triples, URIs, etc.) from the following resources: (a) the dump of the data which were used in [11], (b) online datasets from `datahub.io` website and (c) a subset of *DBpedia* version 3.9. To test the algorithms in even bigger datasets we managed to find metadata from *datahub.io* for 600 datasets of various domains. Comparing to the 287 datasets (see Figure 1(right)), for most of these 600 datasets we were not able to access and download their content (i.e., triples, URIs, etc.). However, we managed to find some basic metadata for these datasets in *datahub.io*. Unfortunately, in *datahub.io* there is a lack of information for other features of these datasets such as the number of URIs, literals, blank nodes and degree of URIs. Therefore, it is not possible to produce feature-based buildings for these datasets, although the proposed visualizations can support feature-based buildings for thousands of datasets. Figure 5 shows on the left side the cyclic spiral layout and on the right side the orthogonal layout for this set of 600 datasets.

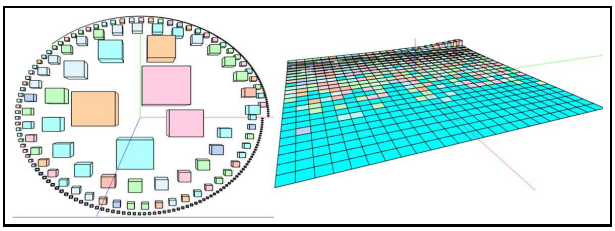

Figure 5: 3D Visualizations of 600 datasets

4 IMPLEMENTATION AND FUTURE STEPS

We have implemented a web-based visualization system, which could be easily accessible by any user. We used the JavaScript library `Three.js`[4] which in turn uses the `WebGL` API[5], which is widely supported by all modern desktop and mobile browsers without the use of plugins. `Three.js` offers a less tedious programming environment in comparison to WebGL, by abstracting away many of the WebGL details, which is a JavaScript API that allows the creation of GPU accelerated 3D graphics and animations inside the environment of a web browser [2] [6].

Figure 6 shows an overview of the web-based visualization system. The visualization is interactive, allowing the user to zoom in any part of the model. For instance, one can change the perspective, the shape of the buildings or their placement, search for a dataset through an *auto-completion search*, see all the connections or those of one dataset, and others. The presented model could be improved in several ways. Below we sketch two indicative enrichments: (a) for aiding the user to get a more informative and "live" overview immediately the system could be enriched with "guided tours" , i.e. with trails of camera movements over the space occupied by the buildings and (b) for reducing the crossings of the edges, each set of buildings that forms a strongly connected component could be visualized as a small round park (or roundabout) where only one line segment connects each building to that park.

[4] three.js is available at http://threejs.org/
[5] https://www.khronos.org/webgl/
[6] There are similar JavaScript libraries like GLGE, SceneJS, PhiloGL, etc.

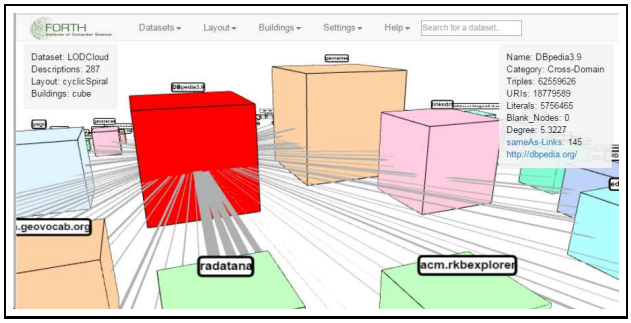

Figure 6: Overview of the web GUI of the system

5 CONCLUDING REMARKS

The proposed 3D interactive system: (i) illustrates accurately the relative sizes of the datasets in triples, (ii) can indicate the average degree of the datasets, (iii) allows the user to control which connections to show or hide, (iv) makes evident (through the layout algorithms) the differences in the sizes of datasets or their commonalities. It supports various building types (cubes, context-dependent cuboids, feature-based cuboids), as well as several layout algorithms(mountainside, orthogonal spiral, cyclicSpiral, similarity-based adaptations of force-directed algorithms), that order the buildings appropriately, depending on the user needs, and similarity-based adaptations of force-directed algorithms.

Acknowledgements. Work partially supported by a) the EU project BlueBRIDGE (Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth), H2020-EINFRA-2015-1, 2015-2018 and b) the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI).

REFERENCES

[1] Aba-Sah Dadzie and Matthew Rowe. 2011. Approaches to visualising Linked Data: A survey. *Semantic Web* 2, 2 (2011), 89–124. http://dblp.uni-trier.de/db/journals/semweb/semweb2.html#DadzieR11
[2] Brian Danchilla. 2012. Three.js Framework. In *Beginning WebGL for HTML5*. Apress, 173–203. https://doi.org/10.1007/978-1-4302-3997-0_7
[3] Javier D Fernández, Miguel A Martínez-Prieto, Pablo de la Fuente Redondo, and Claudio Gutiérrez. 2017. Characterising RDF data sets. *Journal of Information Science* (2017).
[4] Thomas MJ Fruchterman and Edward M Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and experience* 21, 11 (1991), 1129–1164.
[5] James Hendler. 2014. Data Integration for Heterogenous Datasets. *Big data* 2, 4 (2014), 205–215.
[6] Shah Khusro, Fouzia Jabeen, Syed Rahman Mashwani, and Iftikhar Alam. 2014. Linked open data: towards the realization of semantic web-a review. *Indian Journal of Science and Technology* 7, 6 (2014), 745–764.
[7] Jakub Klimek, Jiri Helmich, and Martin Necasky. 2015. Use Cases for Linked Data Visualization Model. In *Proceedings of the Workshop on Linked Data on the Web (LDOW) (CEUR Workshop Proceedings)*. Aachen. http://ceur-ws.org/Vol-1409/#paper-08
[8] Luca Matteis. 2014. VoID-graph: Visualize Linked Datasets on the Web. *CoRR* abs/1408.6691 (2014). http://arxiv.org/abs/1408.6691
[9] Michalis Mountantonakis and Yannis Tzitzikas. 2016. On Measuring the Lattice of Commonalities Among Several Linked Datasets. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1101–1112.
[10] Peng Peng, Lei Zou, M Tamer Özsu, Lei Chen, and Dongyan Zhao. 2015. Processing SPARQL queries over distributed RDF graphs. *The VLDB Journal* (2015), 1–26.
[11] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. 2014. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC 2014*. Springer, 245–260.