RDF Visualization using a Three-Dimensional Adjacency Matrix

Mario Arias Gallego
Computer Science Dept.
Univ. of Valladolid, Spain
mario.arias@gmail.com

Javier D. Fernández
Computer Science Dept.
Univ. of Valladolid, Spain
jfergar@infor.uva.es

Miguel A. Martínez-Prieto
Computer Science Dept.
Univ. of Valladolid, Spain &
Univ. of Chile, Chile
migumar2@infor.uva.es

Pablo de la Fuente
Compt Science Dept
Univ. of Valladolid, Spain
pfuente@infor.uva.es

ABSTRACT

Previous RDF visualization tools generally use node-link representations of the RDF graph to visualize its information. This approach may be enough for small data sets, but it becomes unmanageable as the number of triples increases. Despite advanced node-merging and layout algorithms exist, their outcome do not provide a clear view of the overall RDF structure. In this paper we propose using a 3D adjacency matrix as an alternate visualization method for RDF. We first accurately describe how to graphically organize and annotate the triples. Then, we provide some insights on how to interpret the visualization method by analyzing a real-world RDF data set. Finally, we describe some of the most common patterns found in RDF data.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Human Factors, Design

Keywords

RDF Visualization, adjacency matrix, RDF analysis

1. INTRODUCTION

The RDF[1] W3C Recommendation, provides a simple declarative data model of statements (subject, predicate, object) to describe resources. A set of these triples can be seen as a graph of knowledge, where different resources are described using properties and linked together. RDF itself is schema-relaxed, the vocabulary evolves as needed on demand and it is described in RDF itself. This allows interconnecting heterogeneous sources of data but makes the structure complex due to its fine grain.

Despite the fact that RDF was originally devised for the Semantic Web, an increasingly number of RDF data sets are being published from diverse areas of application such as bioinformatics, social networks, geographic locations, books or films. The *Linked Data Project*[2] has emerged as an initiative to promote the use of RDF to publish structured data on the Web in a distributed and interconnected manner [1]. Latest Linked Open Data (LOD) cloud estimations[3] show that more than 25 billion RDF triples are being shared and increasingly interconnected (close to 1 billion links).

RDF data tend to be large, complex, and hard to read in its textual format. Hence, semantic information developers need visual tools to help them understand its content. Some of the typical tasks are identifying which are the most relevant resources in the graph, which are the most common literals, whether the information is grouped or scattered, count the number of elements of each type, or see the links between resources. Finding an adequate way of visualizing this data is an excelent tool for the developers to know in a straightforward manner how the information is distributed within the data set.

Previous works usually generate a node-link graph to represent the associated RDF graph on screen. While this approach is valid for small data sets, the result is too confusing for large data sets containing millions of triples, and consequently tons of nodes and relationships. A solution to this problem is focusing on a subgraph composed by all nodes within a constant graph-traverse range respect to a center node. This way, the visualization provides more detail of the specified subgraph, but disregards the remaining information and therefore loses the overall view of the data.

We propose using a 3D adjacency matrix as an alternate visualization method for RDF data sets. This approach complements previous works providing a different view of the information, which overlooks fine-grain details to concentrate on the lattice itself. We think that this visualization method

[1] `http://www.w3.org/TR/REC-rdf-syntax/`

[2] `http://linkeddata.org`

[3] `http://www4.wiwiss.fu-berlin.de/lodcloud/`

SEMSEARCH2011 Hyderabad, India

shall be useful to semantic data developers, RDF store and search engine designers to understand the peculiarities of the different data sets.

In section 2, we review the state of the art in RDF visualization. Then, we thoroughly describe our visualization method in section 3. Next, in section 4 we proceed to use our tool to analyze a real-world data set as an example, providing some insights on how to interpret the visualized information and which are the most common patterns found in RDF data. Finally, section 5 summarizes our conclusions and future work.

2. PREVIOUS WORKS

RDF is not meant to be read by humans but by computers. However, it is important that those who generate or consume information in this format can also browse and understand it. Several authors have addressed this problem and have proposed solutions to visualize RDF data, providing valuable tools to simplify this task to the user.

One of the most common ways of facing RDF visualization is considering RDF as a big graph, and then use traditional large-graph node-link visualization techniques, like those used on Web graphs, social networks, or DNA microarrays. The biggest issue of this approach is scalability due to the vast size of the data. Since RDF data sets contain thousands to millions of statements connecting tons of resources, there is a huge number of graph nodes, edges, and a numerous amount of high-degree nodes [2, 3]. As the number of nodes increases, the *readability issue* becomes even more noticeable. Due to the big amount of information, the users begin to have trouble identifying the single elements and the visualization looks like an unrecognizable smudge.

There are several proposals to alleviate the readability problem on big node-link graph visualizations. Their main purpose is reducing the amount of shown information to only present one facet, easily understandable by the users. For instance, node-merging techniques join similar or related nodes into clusters that will appear as a single element in the visualized graph, therefore hiding the complexity of its internals [4]. The cluster node might also contain a summary of its contents. For example, in a RDF database of people, we could gather every triple referring to the same individual into a single node, label it with the URI, and include a list of the properties of that specific person, like the full name, Web page and email address.

Another possibility is allowing users to disclose the details of a cluster by using the *expand* operation or merge several nodes into a cluster using the *contract* primitive. This way we can construct a hierachically structured graph and let the user browse it by selecting which cluster she is interested in. Afterwards, we can show another view which summarizes the nodes the user have traversed, using an method called *triangle-layout* or incremental exploration. The user starts from an initial center node, and the system constructs a tree which includes all graph edges when the user traverses them for the first time, generating a so-called *navigation tree*. Then a traditional triangle-layout algorithm can be used to render the tree [4].

Despite RDF can be seen as a big graph, and this representation is advantageous to perform many tasks, some authors [8] highlight that the graph representation is not convenient to perform every single activity regarding RDF, namely visualization. They argue that although graph visualization might seem the natural extension when the data is already in graph format, we must take into account which is the task being pursued, which is the question being answered, and which are the most interesting aspects of the information in that specific case. For instance, we shall not use the same visualization technique to analyze context/densities than to calculate paths or identify hub nodes.

A completely different approach of rendering graph data is using its adjacency matrix. It consists of generating a boolean-valued connectivity table where rows and columns represent the vertices of the graph, and each cell (x,y) states whether x is connected to y or not. Then we render the table coloring each entry using two different colors for each boolean value.

Previous authors [5] compare traditional node-link against adjacency matrix visualization techniques. The advantage of node-link is that people is more familiar to this kind of representation, consequently, they understand them easily. Its biggest drawback is the problem of node overlapping and edge crossing that result in complex structures that are not easily identifiable. However, this problem has been fairly studied and advanced layout and clustering algorithms exist. On the contrary, adjacency matrix techniques do not manifest the node overlapping and edge crossing issues, since there is no need for layout whatsoever. The most common tasks of graph visualization can be directly performed against the adjacency matrix itself except for path traversals, in which node-link is more natural [5]. Nevertheless, users are not used to perform these tasks using matrices and need a before-hand training to undertand its contents. In general, the node-link approach is more effective for small and sparse networks, whereas matrices are more suitable when the graph is big and dense [5].

Some examples of real-world software implementing the previously described ideas of node-link representations are IsaViz[4], InfoVis[5], VisualRDF[6] and RDF Gravity[7].

To the best of our knowledge no other works use matrices to visually represent RDF data. However, there are other areas of study where these kind of visualizations have been successfully applied, for example social networks [7].

3. ADJACENCY MATRIX VISUALIZATION

We propose an RDF visualization tool based on a 3D adjacency matrix of subjects, predicates and objects. This representation provides a more intuitive view of the structure and distribution of RDF statements within the data set, no matter the magnitude of the data.

We start by converting the plain RDF data into a compact representation [6], so that the information is more easily manageable. The first step is assigning a unique numeric ID to each different subject, predicate and object. To do so, we construct a dictionary that matches strings to IDs and vice versa. We split the dictionary in three different areas depending whether the string appeared as a subject, predicate or object. Moreover, we add a fourth area where we include all those strings that appear both as a subject in at least one triple, and as an object in another one (See

[4] http://www.w3.org/2001/11/IsaViz/
[5] http://ivtk.sourceforge.net
[6] http://visualrdf.sourceforge.net/
[7] http://semweb.salzburgresearch.at/apps/rdf-gravity/

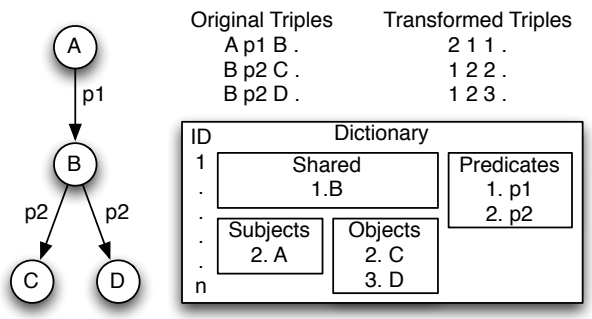

Figure 1: Example of dictionary areas and their ID assignations for a simple RDF graph.

Figure 1). Making this distinction is really important, because this shared subject-object area represents the links between RDF resources. We sort each dictionary block in lexicographic order, and then we sequentially assign numerical IDs to each entry, leaving the first indices to the shared area and then continuing with the subjects and objects, as depicted in Figure 1. Note that IDs are not globally unique, they depend on the area that they apply to. For instance, ID 2 will match to a different string depending whether we refer to the second predicate or the second subject.

Thereafter, instead of dealing with RDF statements containing long strings, we just need to represent them as triples composed by three integers referring to entries in the dictionary. This triple (s, p, o) can be seen as a (x, y, z) coordinate in a 3D space that can be plotted as point in a 3D scattered plot. We let the y axis represent subjects, the x axis objects and the z axis predicates. We use OpenGL[8] to be able to render on screen a huge amount of points by using the hardware-acceleration facilities of the graphics card.

Since we are using a two-dimensional output device, such as a screen or paper, we need a projection to reduce these 3D points into a 2D space. We use an orthogonal projection, so all points are aligned to the axis and hence, they are not affected by any perspective correction. Given that we are using OpenGL as renderer, we can let the user rotate and zoom the view to have different 3D perspectives of the data. The first and most interesting view is the one that places the camera on the z axis looking at the origin, therefore showing a 2D figure comparing subjects against objects (Figure 2). In order not to dismiss predicates, we color them using a highly-contrasted color wheel.

Each axis scale is annotated using the IDs, so the user can get a first sight of the amount of subjects and objects. Since we are using a lexicographic order on each axis, all URIs, blank nodes and literals are grouped due to the leading characters `<"_`. We can also distinguish the shared subject-object area of the dictionary, the rectangle at the origin highlighted using a different background color. This area is quite interesting indeed, because it represents the links among RDF resources. For instance, the proportion of this area respect to the total figure expresses the connectivity of the RDF graph. If this shared area is huge, it means that the data set mainly depicts resources and connections among them. Conversely, when this area is small it suggests that the RDF resources are barely interconnected, and the data set mostly describes properties of several isolated entities. This area is also interesting from the RDF store and search point of view because it represents the elements involved in subject-object joins, which are very common in SPARQL queries.

Although we use a compact representation of the RDF graph and OpenGL to enhance performance, scalability is still an issue when dealing with huge RDF data sets such as DBPedia[9], which contains around 1 billion triples. Transfering that amount of points to the graphics card would be really slow, if feasible at all. We consider that many are redundant, and therefore can be directly dismissed. We decided to use a very simple systematic sampling approach to keep only every k-th triple of the collection, and discard all the rest. We found that rendering $400k$ points was a reasonable compromise between frame rate and detail for our test machine[10]. Moreover, we discovered that by reducing the number of rendered points we improve the occlusion problem of some points being hidden behind others.

We can enhance user experience by providing some extra features for interactively browsing the data. Besides zoom and free 3D rotation, we let the user hover the mouse above the graphic, showing details of the nearest triple under the cursor. To do so we just need to perform the inverse of the modelview transformation matrix and obtain a (s, p, o) coordinate. Then, we search the nearest triple in the dataset compared to the cursor position in terms of euclidean distance, and we regenerate the full triple as string using the dictionary. To perform this operation efficiently, we keep the triples ordered and access them using binary search.

4. ANALYSIS

In this section we show how to use our proposal to browse real-world RDF data sets and interpret its features. We use the Billion Triple 2010 data set[11] from the Semantic Web Challenge. Since it is too big, we first split it into more manageable pieces. We use the host available in the fourth provenance component of the NQUAD format, obtaining as a result several RDF data sets of different sizes. Then, we perform an exploratory analisis of the data using our tool, seeking which are the most common patterns and how to interpret them against the underlying data.

Figure 2 shows the Bibsonomy data set adjacency matrix as browsed using our tool. To put the reader in context, Bibsonomy is a social bookmarking and scientific publication sharing system that allows users to share online references to their favourite documents with other groups of people within the community. We choose it because it contains several of the most significative patterns found in RDF data.

The first thing that comes to our attention is that we can identify regular shapes. We can see on top of the figure that it contains 13 million triples and 409 different predicates, being a considerably big data set. If we check the labels of each axis we notice that it contains 3.6 million subjects and 5 million objects, being the proportion of objects bigger than the subjects. If we obvserve the shared area on the left we see that it occuppies an important proportion of the whole figure. This fact indicates that there is a fair amount

[8] http://www.opengl.org/

[9] http://dbpedia.org

[10] Macbook Core2Duo 2Ghz. 4Gb RAM. GeForce 9400M.

[11] http://km.aifb.kit.edu/projects/btc-2010/

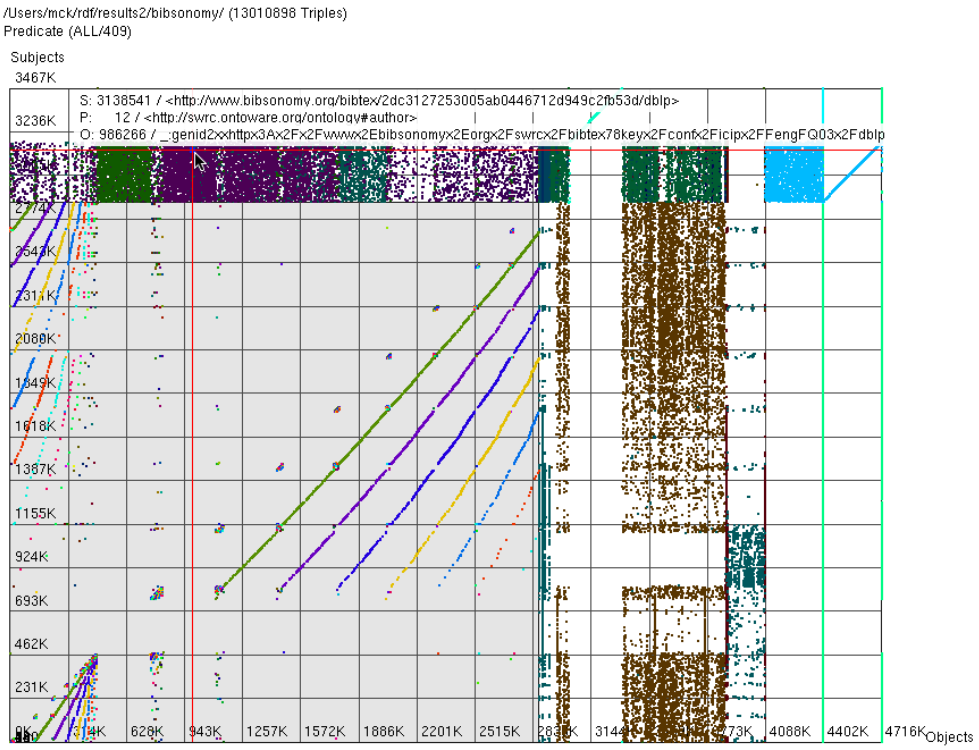

Figure 2: Bibsonomy data set as shown using our visualization approach.

of connections among resources in the graph.

Then, we can start hovering the mouse on the figure to check what kind of information contains each of the areas of the graph. We start with the shared area and we notice that all the appearing lines are just blank nodes used to define RDF lists using the RDFs container membership property `rdfs:_1`, `rdfs:_2`, and so forth. Blank nodes are intermediate nodes employed to link other resources together (as seen in the lists example), therefore, they usually appear in the shared area.

If we check the green and purple areas on top of the shared area, we see the definitions of authors and editors linked to the previously mentioned blank nodes. If we browse the big brown area on the right, we find out that it gathers all the names of the authors of the documents. Also very important are the light-green vertical lines on the right of the figure. They represent `rdf:type` defining the class of each subject. As we see, we can analyze each of the areas and understand what kinds of resources and links exist on the data set.

While browsing the different data sets of BT2010, we found the following patterns:

- **Vertical line**. Represents an object that is related to many subjects. It is very common on many data sets, for instance when using `rdf:type` as shown above. Other examples include instances of classes (many subjects related to the type `foaf:Person`) or commonly used literals such as dates. We would like to note, that these lines can be thick, *i.e.* there is a group of objects with common prefix (and therefore close within the x axis) that are related to many subjects.

- **Diagonal line**. They appear when there is a correlation between the subject and object strings. For instance, many URIs from blogs include the date as part of the URI, and also contain a `date` property specifying the date again. It is also very common when resources have numerical correlative identifiers included in the URI. We noticed that whenever duplicated or missing values are present the line is not perfectly straight.

- **Scattered rectangle**. It appears when an interval of subjects is related to an interval of objects. For instance, we may have a set of subjects of the class `Event` that are grouped together because their URIs share a common prefix. They are associated to literals representing dates, which also share a common prefix specifying the year. Therefore these relationships can be enclosed within a square.

Note that there also exist horizontal lines, but they are very scarce. Normally there are no hub subjects that are related to many different objects. It can happen, for instance, when we have big lists gathering many resources of the RDF data.

Another important result when analyzing RDF data sets using the adjacency matrix approach, is that we find signs of locality of reference. All the previously described patterns are evidences that there is a strong correlation between the elements in those areas. This fact is crucial when designing compression schemas for RDF, constructing indices, designing other kind of visualization techniques or devising new semantic search engines.

The adjacency matrix approach is also valuable to evaluate the complexity of RDF structures. In the bibsonomy example, we observe that even though it is a very big dataset,

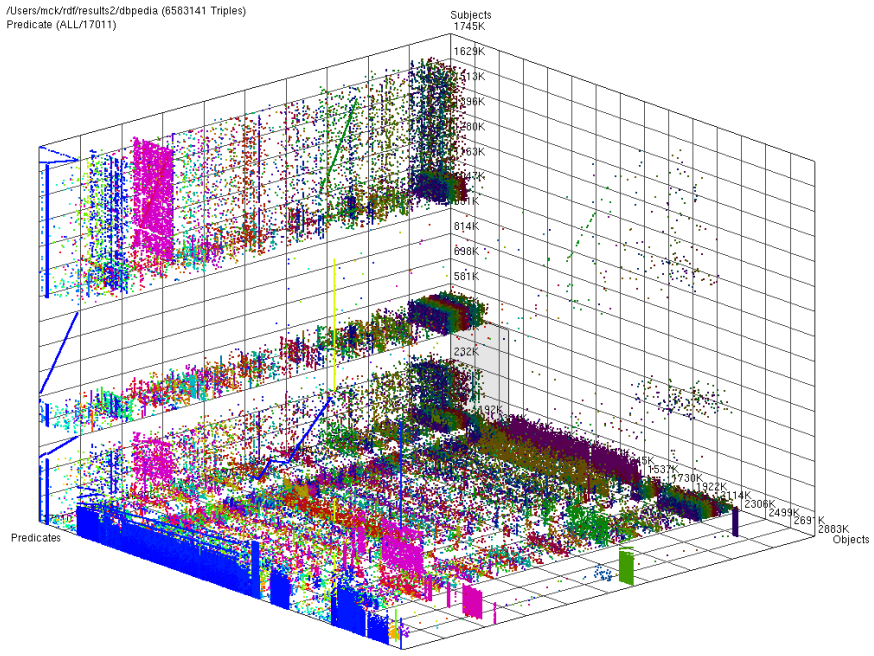

Figure 3: BT2010 subset of the DBPedida data set in 3D view.

its internal schema is quite simple. On the other hand, more diverse data sets like DBPedia, are much more complex as shown in Figure 3.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we face the problem of visualizing large-scale RDF data. We propose a new method based on the RDF adjacency matrix to alleviate the limitations of previous node-link graph visualization approaches.

The major strength of our approach is that it is able to deal with huge data sets with millions of statements and yet provides a means to understand the overall structure of the graph. We used an example to show how to interpret the different pattens that appear in the plot and we provided a possible explanation of why these patterns occur in terms of RDF statements. We showed that this method does not substitute, but complements previous visualization approaches.

We would like to highlight that using our visualization method, we discovered evidences that many RDF data sets present locality of reference. This assumption is crucial when designing RDF stores and search engines.

Our approach might be useful to any user wanting to understand internal structures of RDF data sets, namely semantic content generators, RDF data-mining analysts or RDF store designers. It also serves as a basis to fine-tune existing solutions.

There is much room for future works based on our approach. Two areas of study that we deem particularly interesting are trying other dictionary identifier assignment algorithms besides lexicographic, and applying clustering algorithms to automatically detect regular patterns.

6. ACKNOWLEDGMENTS

Funded by the MICINN (TIN2009-14009-C02-02) and the Millennium Institute for Cell Dynamics and Biotechnology (ICDB) (Grant ICM P05-001-F). The second author is granted by a fellowship from the Regional Government of Castilla y Leon (Spain) and the European Social Fund.

7. REFERENCES

[1] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web LDOW2008. In *WWW08*, pages 1265–1266, 2008.

[2] J. Dokulil and J. Katreniakova. Visualization of Large Schemaless RDF Data. *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 243–248, 2007.

[3] J. Dokulil and J. Katreniakova. RDF Visualization - Thinking Big. *2009 20th International Workshop on Database and Expert Systems Application*, pages 459–463, 2009.

[4] J. Dokulil and J. Katreniakova. Using Clusters in RDF Visualization. *2009 Third International Conference on Advances in Semantic Processing*, pages 62–66, 2009.

[5] J.-D. Fekete. Visualizing networks using adjacency matrices: Progresses and challenges. *2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics*, pages 636–638, 2009.

[6] J. D. Fernández, M. A. Martínez-Prieto, and C. Gutierrez. Compact representation of large RDF data sets for publishing and exchange. In *Proceedings of the 9th international semantic web conference on The semantic web*, ISWC'10, pages 193–208, 2010.

[7] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: a hybrid visualization of social networks. *IEEE transactions on visualization and computer graphics*, pages 1302–9, 2007.

[8] D. Karger and M. Schraefel. The Pathetic Fallacy of RDF. *ISWC06*, 2006.