18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

RDF Versus Attributed Graphs: The War for the Best Graph Representation

Michael Margitus
Information Fusion Group
CUBRC Inc.
Buffalo, New York, U.S.A.

Gregory Tauer
Information Fusion Group
CUBRC Inc.
Buffalo, New York, U.S.A.

Moises Sudit
Information Fusion Group
CUBRC Inc.
Buffalo, New York, U.S.A.

Abstract – In this work, methods are developed to overcome the inherent problems of network abstraction and analysis from multiple heterogeneous data sources. RDF and attributed graphs are two common choices for graph modeling. While both are very similar with respect to the type of information that can be represented, characteristics intrinsic to each representation affect the analysis performed over the resultant network abstractions. By selecting a dual graph representation approach and leveraging the strengths of both models, the semantic analysis performed over RDF graphs is combined with the topological analysis applied to attributed graphs, to produce a comprehensive foundation for network analysis that cannot be easily achieved, nor its value matched, by one representation independently.

Keywords: Graphs, RDF, attributed graphs, network abstraction, data alignment.

1 Introduction

Networks are quickly becoming the de facto representation choice for modeling real world data involving entities and relationships. A network model provides more flexibility over traditional relational data models, and does not require a predefined schema or consistent data keys across each entry of the same type. These criteria are especially important when modeling data that evolves over time, data containing complex relations, or data that has the potential for a multitude of query perspectives.

The process of creating a network abstraction, and analyzing the resultant network is challenging for a number of reasons. Representing entities and relationships from raw data sources as a network in a pragmatic manner; the inability to capture details found in descriptive data through a network representation; and the process of normalizing data to a common structure across data sets or related networks are all factors that, if done poorly, may result in an incomplete and inaccurate analysis network, ultimately leading to incorrect conclusions and results.

Given the increased amount of disparate data required to make timely and informed decisions, a plethora of research is being conducted in the areas of network abstraction and analysis. At the forefront of this area is the objective to produce automated methods for aligning and constructing a network model from heterogeneous, yet related, data sets in order to assess and harvest information.

The choice of network representation is an important decision. Networks are ubiquitously represented as graphs, a finite nonempty set of objects called vertices, together with a set of pairs of distinct vertices called edges [1], however the structure of these graphs, and the varied ways a single data set can be represented as a graph leaves much to consider. Two popular graph topologies are the representation provided by utilizing the Resource Description Framework (RDF) data model, and graphs produced by following an attributed graph model. Although each format can represent the same information, each also has inherent advantages and disadvantages when used as the underpinnings of network analysis.

In this work, we develop methods to address the inherent problems of network abstraction and analysis from real-world data by using both representations. We begin by building an RDF knowledge graph that provides a complete and detailed representation of the data sets under investigation. The RDF representation is well suited to this task since it is expressive, closely linked to knowledge representation tools, and standards based. Unfortunately, the resulting RDF representation can be poorly suited to perform traditional network analysis due to its structure. To overcome this shortcoming, we abstract the RDF graph to an attributed graph through an approach that uses the SPARQL query language in an extraction process, capturing the key entities, events, and relationships relevant to analysis. We then describe the application of graph and social network analytics on the attributed graph abstraction, which cannot be directly applied to the RDF knowledge graph, to derive valuable results concerning the entities and events under observation.

In the remainder of this paper, we discuss each of the graph models in detail along with our rational for using RDF for the master knowledge graph and attributed graphs for the network abstractions that we perform analysis on. (Section 2). Section three describes related approaches

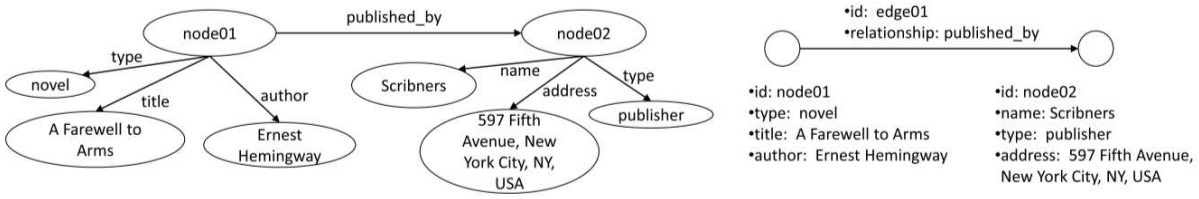

Figure 1. Semantic content represented as an RDF graph (left), and an attributed graph (right).

and how they differ from our strategy. In section 4, we discuss our approach to perform network abstraction and analysis utilizing both RDF and attributed graph formats. Section 5 concludes our discussion with observations, findings and recommendations.

2 Graph abstraction models

In this section we present two common graph representation paradigms, RDF and attributed graphs. Although both graph formats can represent the same content each format has its own strengths to offer.

2.1 RDF

RDF is a model of assigning named properties and values to objects [2]. The RDF model contains 3 sets: Resources, Literals, and Statements. Each statement is a triple of {subject, predicate, object} where a predicate is a special type of resource. Each statement defines a triple that represents a single directed edge within an RDF graph, with the subject (a resource) representing the source vertex, the property representing an edge label, and the object (either a resource or literal value) representing the target vertex. In this way, RDF statements express directed labeled graphs [2,3].

Importantly, the subject of a statement must be a resource, while the object may be either a resource or a literal value (such as a string, integer, date, etc.). Literal values may not be the origin of statements, which allows for literal statements in RDF to model vertex attributes from attributed graphs [4].

RDF graphs excel in conceptual modeling and knowledge representation applications where the expressivity of the format can be leveraged to describe complex relationships and characteristics among existing resources. When concept representation is formalized, for example by an ontology, data can be represented according to the constructs outlined to produce a normalized RDF graph model. RDF is a popular format for ontological modeling, in part due to the designation of the Web Ontology Language (OWL) as a W3C standard [5].

RDF enables analysis to be performed by exploring the model through query interactions. This exploration is made more powerful through the use of an ontology, or a composition of ontologies, to model the objects and relationships found within the data. This facilitates network analysis from an information retrieval perspective. The standard RDF query language is SPARQL [6]. The SPARQL query language for retrieving information from RDF data seeks to find a subgraph pattern match given a pattern defined by the query, and the RDF data graph to search over. Through SPARQL, specific patterns of interest can be discovered. Although a powerful query tool, subgraph matching itself gives limited insight into the overall structure of the graph, including the global importance/centrality of entities.

We focus on RDF models of large and expressive knowledge graphs. The verbose structure of this type of graph makes analysis by classical graph analysis algorithms challenging. This is especially true when data is expressed following a detailed representation. In knowledge representation graphs, the vertices defined can represent a wide variety of objects, not just the types of

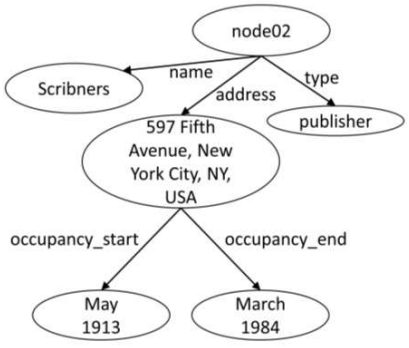

Figure 2. Attributes of attributes, represented in an RDF graph.

Table 1 : Comparison of RDF and Attributed Graphs

RDF	Attributed Graphs
The RDF model requires only triples to be stored.	Objects to be stored depend on the specific implementation.
RDF is a W3C standard, including supporting tools such as SPARQL and OWL.	There is no well-established standards committee resulting in multiple implementations.
RDF clearly defines a set of exchange formats that all databases are expected to support.	It is not easy to transfer data between databases without support for a common exchange format.
Since RDF is a standard, all API implementations follow the same basic model.	Multiple implementations results in confusion between the model, API, and database.
RDF only supports directed graphs.	Attributed graphs can support both directed and undirected graphs.
Edges in RDF only support a single label (the predicate). In some cases, this can lead to complex reification schemes to make statements about a statement.	Edges in an attributed graph can have attributes just like vertices. This allows for the representation of information about the relationships.
Attribute information is stored as triples, leading to more triples than an equivalent attributed graph representation.	Attribute information can be efficiently stored as key/value pairs on the graph elements.

objects that a given graph analytic is interested in. Without filtering criteria at the analytic level, or an extraction of the desired subgraph of vertex and relationship types, all vertex types will be considered by automated analysis algorithms. If, for example, there is a desire to calculate betweenness centrality across all entities of a given type, applying a traditional implementation of the algorithm directly to the RDF knowledge graph will result in unintended properties, entities, and relationships being included within the calculation. This will impact the interpretation of the results. To avoid this side effect, a filtering technique could be applied to extract only the desired entities and relationship types. If done directly on the knowledge graph, this risks a loss of information from the removal of properties/attributes which may be required during further analysis.

2.2 Attributed graphs

An attributed graph $G = \{V, E, A_v, A_e\}$ is defined by a nonempty set of vertices V, a set of edges E (directed, undirected, or a combination), a set of attributes on the vertices A_v, and a set of attributes on the edges A_e. The attributed data represented within the graph exists separately from the graph structure itself, with attributes present on each vertex and edge. This network representation is in contrast to RDF graphs, where vertices and attributes are treated uniformly, leading to a significant increase in size in the graph topology [7].

We are primarily interested in attributed graphs where the attributes of graph elements are represented as key/value stores. Such graphs are often referred to as "property graphs".

Attributed graph models permit a more condensed representation from a visual, analytical, and traversal standpoint. Although literal statements in RDF achieve a similar goal as attributes in an attributed graph, the attributed representation can efficiently encode the key/value pairs [4]. Figure 1 illustrates two graphs representing the same content. The graph on the left is an RDF representation of novel, its publisher, attributes of each, and the relationship between the two. A total of eight vertices and seven edges are created to represent this information as an RDF graph. In contrast, an attributed graph representation (Figure 1, right) can be constructed with two vertices and a single edge, with each graph element having its own set of attributes to detail the semantic data.

A drawback of the attributed graph model is the risk that it may be used to model information in a way that restricts expressiveness. A good example of this is the problem of representing attributes of attributes, such as a valid time period of an address, cannot be easily represented in the above attributed graph model, but can be achieved in the example RDF graph by adding additional edges and vertices incident and adjacent, respectively, to an address vertex, illustrated in Figure 2.

In general, the attributed graph model tends to promote a condensed, streamlined topology that requires less storage, and is more tailored toward structural graph analysis, such as social network analysis algorithms.

2.3 Summary of Comparison

To summarize the above comparison of RDF and attributed graphs: they can both model the same information in nearly the same way, but each has advantages. A high level overview of these advantages is presented in Table 1. Based on this overview, our recommendation for using RDF vs Attributed Graphs is:

Prefer to use RDF for:

- Large knowledge graphs that require rich expressiveness and a foundational ontology. RDF is standards based which is important for knowledge graphs since they often need to be shared. There are many tools built around RDF for knowledge modeling (OWL, etc.)
- Graph data that must be queried for specific subgraphs. RDF can be queried using SPARQL, which is a standards based, full featured query language for RDF.

Prefer attributed graphs for:

- Graphs that need to be queried for elements possessing specific attribute values. Attributed graphs efficiently store attribute values as key/value pairs on the graph elements. This brings both efficiency and conceptual advantages for indexing elements by their attributes.
- Graphs that need to be visualized. It is often easier to render attribute information as a table than as vertices and edges.
- Prototyping and simple applications that will not need to be shared. Subjectively, attributed graph models tend to be easier to work with than RDF from an implementation perspective. This can save time when building prototypes and can make applications easier to maintain.

3 Related work

Related work develops a methodology for storing attributed graphs in RDF triple stores [4]. This related work describes how attributes can be expressed as literal statements and edge attributes can be encoded using reification in RDF.

The concept of semantic social network analysis [8] attempts to take social network algorithms and heuristics ordinarily executed over a traditional entity-relation graph and modify those algorithms so that they can be applied to an RDF formatted graph. The goal is to leverage semantic web technologies to merge and exploit the most valuable features from the domains of social network analysis and the semantic web. Algorithms executed over RDF allows for the preservation and possible utilization of the semantic information found within the data, particularly the typed relations that exist, while at the same time providing the ability to analyze the structure of the graph and interaction between entities to gain valuable insight, for example, the most influential player in the network.

Accomplishing the objective of semantic social network analysis requires each social network algorithm or heuristic to be implemented as a SPARQL query. Starting with an RDF social graph, SPARQL extensions are utilized to extract an RDF-subgraph from the RDF based on defined resources or property types. One or more SPARQL queries analogous to traditional SNA metrics are then applied to the subgraph. For a subset of algorithms, post processing is required.

This method has advantages over the strategy of applying SNA to RDF graphs by extracting a raw, i.e. untyped, graph of one or more relationship types from the RDF graph. From the semantic social network analysis perspective, too much knowledge is lost in this extraction which could otherwise be used for filtering and customizing results. In general, extractions of this form reduce the expressivity of the social network representations to simple un-typed graphs [8].

Our dual graph representation approach, discussed in the subsequent section, addresses the issues intrinsic to an untyped graph extraction from RDF. By leveraging attributed graphs, we are able to keep necessary semantic information intact, preserving entity and relationship types, as well as other useful attributes for each vertex and edge in the graph. Additionally, unlike the RDF-based

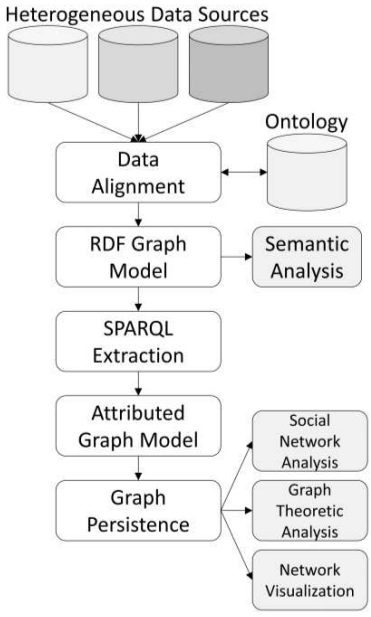

Figure 3. Overview of two graph representation approach.

semantic social network analysis approach, the attributed graph format mirrors a typical entity-relation graph representation, and thus there is no need to create new implementations of traditional SNA centrality and community detection algorithms in order to apply them to the network.

4 A two graph approach

Each of the previously introduced graph formats have the capacity to represent the same, or very similar network and semantic content. Format selection, however, should be a cautious decision dependent on the type of analysis to be performed on the network data, with each graph model possessing its own analytic strengths and weaknesses. Proper structural and semantic network analysis can be more easily achieved by using multiple formats, leveraging the strengths of each format to divide and drive the investigation. To this end, we have chosen a two graph representation approach, sacrificing increased storage costs for an improvement in the flexibility and quality of analysis.

At a high level our approach follows as depicted in Figure 3. Given multiple heterogeneous data sets, we seek to create and analyze a normalized common graphical model. To construct this model, we begin with an ontological alignment process to homogenize all available data from the raw data sources. The product of this process is an all-encompassing RDF knowledge graph, created for two purposes. First, the RDF graph captures all available knowledge at a detailed level and is the foundation for all further analysis. Utilizing the detailed data representation of the aligned RDF graph, SPARQL queries can be issued for targeted information retrieval and initial exploratory analysis. Secondly, this graph acts as the basis for deriving an attributed graph, extracted from the RDF representation to be used for social network, and structural graph theoretic analysis.

Leveraging both graph formats provides complete analytic coverage, and the approach provides a number of advantages over conducting two separate forms of analysis from two independently constructed graph representations. Deriving the attributed graph from the RDF representation promotes consistency between the two models from an analysis perspective. All entities and relationships analyzed in the attributed graph can be traced back to, and analyzed in the RDF graph, without the potential conflicts that may occur if the graphs were constructed independently. Anchoring the attributed graph to the RDF graph also ensures that the data represented in the attributed graph has a standard, widely recognized, model representation. Finally, construction from the RDF representation uniformly establishes semantic information for the attributed graph, provided by the ontology utilized to generate the RDF graph.

4.1 Attributed graph generation

At the core of our approach is a methodology for using SPARQL queries to define a translation from RDF to attributed graphs. To accomplish this translation we define two types of SPARQL queries:

1. Queries that define the attributes of a vertex.
2. Queries that define relationships between vertices.

The system issues all defined SPARQL queries against the RDF store. Every result from a query that defines attributes of a vertex are transformed into an attribute on a vertex. Similarly, every result from a relationship defining query is instantiated as an edge in the attributed graph.

After all queries have completed, the results are made available through an attributed graph API, such as Titan or the Blueprints Graph API [9].

This approach is tested on a dataset generated using the Berlin SPARQL Benchmark generator ("benchmark generator") [10]. This dataset simulates an e-commerce use case with products, product features, reviewers, reviews, producers, and others.

We investigate an abstraction that could allow the development of an algorithm to examine reviewer's preferences in product features. To accomplish this, our abstraction contains three types of entities: reviewers, products, and product features. Reviewers are connected to product features by an edge if they have reviewed that product, and products are connected by an edge to product features if they possess that feature.

The benchmark generator was used to produce a 10,000 product dataset that also had 10,949 product features and 100,000 reviews. Each review was authored by one of 5,152 reviewers. The resulting RDF triple store had 3,576,172 triples.

The desired abstraction requires 5 SPARQL queries:

1. Extract product features as vertices.
2. Extract product names as attributes of product vertices,
3. Extract reviewer names as attributes of reviewer vertices,
4. Create edges from products to their features, and
5. Create edges from reviewers to reviewed products.

After loading the RDF file, it took a total of 14.1 seconds to run the above SPARQL queries and produce the attributed graph abstraction. All processing was done in memory, which was constrained to 3GB. The resulting attributed graph contained 26,101 vertices, 78,303 vertex attributes, and 291,496 edges each with a single attribute.

The attributed graph abstraction improves on the original RDF representation for answering questions on reviewer preference for product features by discarding information that was not necessary (producers, vendors, etc.) as well as by condensing structural information in the graph. The result is a graph that is convenient to work with

when answering specific questions. However, because it is a specialized abstraction, the resulting attributed graph will not be applicable to every question that might be asked of the original data.

4.2 Graph analytics and visualization

For subsequent graph analyses, we choose to persist the attributed graph generated through the SPARQL query extraction process in a Titan graph database [11]. Property keys are created and indexes are defined on the attributes for efficient retrieval during the SNA and graph theoretic analysis stage.

The graph that exists within the graph database can be classified as a specialized case of an attributed graph called a layered multi-modal network [12], containing multiple entity and relationship types from layers of heterogeneous sources. These characteristics allow layered multimodal network analysis (LMMNA), which includes social network and graph theoretic analysis, to be performed through a custom graph analysis and visualization application.

The application provides a set of graph visualization perspectives—graph filtering, timeline progression, and geospatial context visualization—which produce concise, interactive renderings of the information originally represented in the verbose RDF format. Additionally, the application supplies classic and custom graph analytics including centrality measurements, community detection, shortest path analysis, graph matching, graph clustering, and subgraph generation (e.g. *k*-hop neighborhood derivations). Applying these analysis techniques to the expanded topology of the RDF graph would result in inaccurate results, skewed due to extraneous, non-entity type vertices being included in the calculation. For example, a simple degree calculation would report entities with a large number of attributes as being more central in a social network due to these "artificial" connections.

5 Conclusion

Attributed graphs and RDF graphs are very similar with respect to the type of information that can be expressed. The determining factor for graph representation selection should be the type of analysis to be performed on the graph. RDF is a highly expressive semantic data model that, when paired with a detailed ontology, can capture an intricate representation of complex data relations. RDF graphs can be efficiently analyzed through SPARQL queries, defining patterns of interest to a high degree of precision. The RDF model is also more suitable for representing hierarchical data, where a granular breakdown may be necessary. An example of this would be modeling an address, which can be further decomposed into the street, city, state/province/territory, and country parts. Each of these concepts can be represented in an attributed graph as properties of a vertex, however the tiered relationships between these concepts that are present in the RDF format are lost.

The uniform treatment of entities and attributes as vertices in an RDF graph result in a generally larger topology than with an analogous attributed graph representation. For this reason, attributed graphs excel when used for structural graph analysis, where semantic data may be leveraged, but is not heavily relied upon (e.g. used for context, or filtering). The condensed representation is more fitting for classic social network analysis algorithms than the RDF representation, where the presence of ancillary attribute vertices will influence the computation.

Capturing the semantic and topological information in a single graph abstraction is difficult to do well. Our combined approach overcomes these challenges by delegating to each format the tasks for which it is best suited. This strategy has proved initially valuable, providing multiple perspectives to derive comprehensive results concerning entities and events under investigation.

References

[1] G. Chartrand and L. Lesniak, *Graphs & Digraphs*, 4th ed. Boca Raton, FL: Chapman & Hall/CRC, 2005, ch. 1, pp. 1.

[2] *Resource Description Framework (RDF) Model and Syntax Specification* [Online]. Available: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[3] Hayes, Jonathan. "A graph model for RDF." Diploma thesis, Technische Universit¨at Darmstadt Universidad de Chile, 2004.

[4] S. Das et al., "A Tale of Two Graphs: Property Graphs as RDF in Oracle," in Proc. Of 17th International Conference on Extending Database Technology, Athens, Greece, 2014.

[5] *OWL: Semantic Web Standards* [Online]. Available: http://www.w3.org/2001/sw/wiki/OWL

[6] E. Prud'hommeaux, A. Seaborne. (2008) "SPARQL Query Language for RDF" [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/

[7] S. Sakr, S. Elnikety, and Y. He. "G-SPARQL: A Hybrid Engine for Querying Large Attributed Graphs," *In CIKM '12*, Oct 29 – Nov 2.

[8] Ereteo, G., Gandon, F. L., Corby, O., Buffa, M.: "Semantic social network analysis"; CoRR; abs/0904.3701 (2009).

[9] *The benefits of Blueprints* [Online]. Available: https://github.com/tinkerpop/blueprints/wiki/The-Benefits-of-Blueprints

[10] C. Bizer and A. Schultz. "The berlin sparql benchmark". *International Journal on Semantic Web and Information Systems*, vol. 5, 1-24, 2009.

[11] *Titan: Distributed Graph Database* [Online]. Available: http://thinkaurelius.github.io/titan/

[12] P. LaMonica, T. Waskiewicz, "Layered multi-modal network analysis of textual data for improved situation awareness," *In Proc. International Conference of Knowledge Engineering (IKE),* July 2011.