

An Analytic Approach for Discovery

Eric Dull and Steven P. Reinhardt

Cray, Inc.

{edull,spr}@cray.com

Abstract—With the widespread awareness of Big Data, mission leaders now expect that the data available to their organizations will be immediately relevant to their missions. However, the continuing onslaught of the "data tsunami", with data becoming more diverse, changing nature more quickly, and growing in volume and speed, confounds all but the simplest analysis and the most capable organizations. The core challenge faced by an analyst is to *discover* the most important knowledge in the data. She must overcome potential errors and inaccuracies in the data and explore it, even though she understands it incompletely, guided by her knowledge of the data's domain.

We have solved customer problems by quickly analyzing numerous dimensions of data to check its sanity and to compare it to expected values. Guided by what we find initially, we quickly move on to further (unanticipated) dimensions of the data; discovery depends on this ability. This approach vitally brings the analyst into direct interaction with the data.

We implement this approach by exploiting the ability of graphs (vertices and edges) to represent highly heterogeneous data. We use RDF as the data representation and SPARQL for queries. We use non-parametric tests, which are readily targeted to any type of data. This graph-analytic approach, using proven techniques with widely diverse data, represents a guidepost for delivering greatly improved analysis on much more complex data.

I. INTRODUCTION

Analyst groups are being strongly challenged to understand quickly the insights latent in their organization's data, despite its diversity, changing nature, volume, and speed. We focus on the *discovery* aspect of analysis, where the analyst cannot rely on techniques that have run previously on the given datasets, but rather must explore within the data in a way that cannot be predicted. The organization expects that the analyst will quickly uncover the most important knowledge in the data. Using her knowledge of the data's domain, she must explore the data despite its potential imperfections.

If we define *discovery* as finding a connection in the data of which the analyst was previously unaware, it requires more than just delivering existing capabilities a little faster or more easily used. Rather, it requires enabling a subject-matter expert (SME), *i.e.* the person with the most knowledge and intuition about the data, to explore the data quickly, and even, by appropriate pre-analysis, pulling the SME's eye to aspects of the data that are likely to be most fruitful of further exploration. We have evolved an analytic approach to discovery, implemented in the semantic technologies RDF and SPARQL, that enables rapid progress through analytic questions toward an analytic goal. We have solved high-value customer problems by quickly analyzing numerous dimensions of data to check its sanity and to compare it to our expectations, then moving on to further dimensions of the

data guided by what we find initially. Sometimes we analyze data and compare it to our mental expectations. Other times we compare data from one subset (place, time, etc.) to that from another subset, to see if they differ in unexpected ways. Sometimes we analyze the *values* of the data and other times the *connectivity* of the data. Sometimes we create a synthetic grid or discretization, say of geospace and time, to represent data for fast comparison. Discovery depends on being able to compare dimensions quickly, without knowing in advance the dimensions to compare. Bringing the SME into direct interaction with the data is essential to accelerating discovery.

We implement this analytic approach by exploiting the ability of graphs (vertices and edges) to represent the richness of highly heterogeneous data and enable discovery within it. To date, we use RDF as the data representation and SPARQL for queries, queries that can build on each other to focus rapidly on the highest-value knowledge (in the estimation of the subject-matter expert) in the data. (We use *heterogeneous* rather than *semantic*, because this approach is not limited to natural language.) RDF supports complex, dynamic ontologies, though that adds a burden of discovering the current ontology, which we often achieve by summary graphs of vertex-types and the edge-types that connect them. We use Jaccard scoring [6] and non-parametric tests (typically Kolmogorov-Smirnov[7]), which are readily targeted to any type of data when guided by a SME. Other non-parametric tests could easily be used in place of these. RDF and SPARQL are one data format and language that support implementation of this approach, but it may be implemented with other technologies, such as Spark/GraphX from the Berkeley Data Analytic Stack [2].

The graph-analytic approach, using proven techniques with widely diverse data, represents a guidepost for delivering greatly better analysis on much more complex data.

II. ADDRESSING THE CORE CHALLENGE

The core challenge facing many analytic organizations, illustrated in Figure 1, has at its center a data repository, with both new instances of existing types of data and instances of new types of data flowing into it. Results from existing analytics must flow out to meet existing production needs, while new analytics must be created to discover further knowledge in the data. This paper focuses on this discovery process, based on our experience working with a customer needing to understand traffic and vulnerabilities on its corporate network. In that context, some key entities are IP addresses, ports, protocols, and Internet domain names. Section III.B provides an example of these techniques applied to flow-data

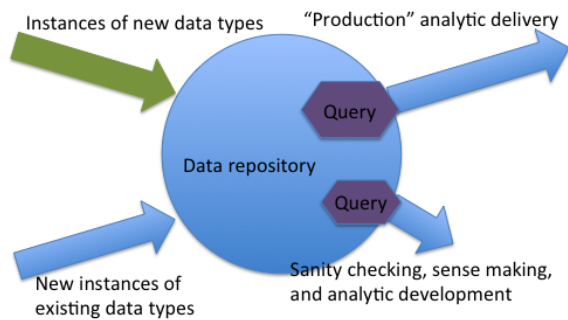


Fig. 1. High-level Workflow

III. ANALYTIC APPROACHES FOR DISCOVERY

We start from the point of view of an analyst with little knowledge about a body of data and proceed to the point of deep knowledge about the data. Many of the techniques we describe are useful at multiple points along this continuum.

Analysis is an iterative process that consists of framing the analytic problem, defining an approach, gathering the applicable data, understanding the biases present in the data (via sanity-checking and sense-making), applying analytic techniques consistent with the approach, determining the analytic results that answer the problem, and documenting the answer to the analytic problem.

A. Sense-making

When an analyst first gets a new corpus of data, even new instances of an existing corpus, the first task is to ascertain the sanity of the data, and then understand it deeply enough to enable further analysis. In practice, an initial sanity-check on data is often necessary, but even then errors in the data may still surface as analysis becomes more precise. For instance, if the data claims to cover the 24 hours of a day, but only has data in the hours 0 through 9, there may have been an error in the software that generates the data. In a heterogeneous context, if flow-data records use IP addresses that are not found in the firewall records, it may be that the data is truly disjoint, or it may be that the IP addresses in the flow data and firewall data have not been constructed the same and hence do not match. In either case, the anomaly needs to be understood before continuing with the data.

With heterogeneous data, sense-making includes discovering the *ontology* (or schema, in relational database terms) in which the data is represented, as that ontology is not explicitly defined anywhere [1]. A *summary graph* depiction of the ontology is shown in Fig. 2, with vertices representing the types of objects in the data. The edges summarize the edges between instances of the types of objects that are the subject and object of triples in the data; edge thickness represents the number of edges between the two types. While this figure is simpler than many real-world ontologies, it illustrates how the structure of the data can be readily represented for to aid an analyst’s understanding. Note that the visualization of

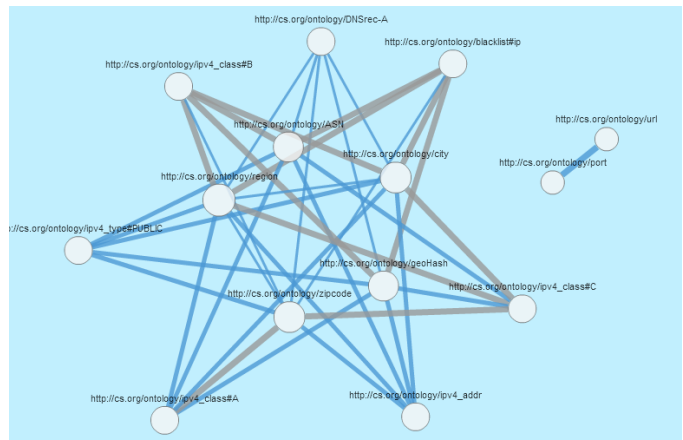


Fig. 2. Discovered Ontology

the ontology as a graph is natural; whether the underlying representation is graph-based or not is immaterial at this level.

A next level of sense-making analyzes the values of a given type of edge. For instance, flow data is often discretized such that a flow cannot last more than an hour. Similarly, TCP/UDP ports are limited to 0-65535. If the data has values outside those ranges, it is suspect. (Additionally, the analyst must ensure that the right units (*e.g.*, English v. metric) are being used.) Not all types of edges will have values that will be readily known by the analyst, but for those that are, this can be a simple way of surfacing the subject-matter knowledge of the expert, by showing the values in the data and letting the analyst respond if the data looks suspect. Beyond correctness, understanding the semantic meaning of a field, *e.g.*, a timestamp, is important. What is the format of the timestamp (absolute, relative)? What is its precision? When was it collected? What specific event is denoted by the timestamp?

Once the basic structure of the data and its values are understood, other questions arise, such as the quantity of the data and whether that indicates that we have all the data we expected or not. *E.g.*, if the data purports to capture all flow data from a 24-hour period, and we know there are about 30,000 flows per second, we should have about 2.6 billion flows. If we do not, we may be missing data, the data may be summarized, or there is some other effect; in any case, the analyst needs to understand the anomaly before proceeding.

Understanding the data values in more detail is another part of sense-making. What are the dominant moments, and, often as importantly, what are the outliers? Analysts know that real-world data is noisy and messy, so will want to avoid actual noise but at the same time want to understand rare but real events accurately. In addition to looking at the total data, we often gain insight by comparing data from entities selected from different ranges in some important dimension, like place or time, to see whether the values in a given dimension also vary; *e.g.*, are the top N most frequent external domains in Internet flow data from the day before yesterday and yesterday

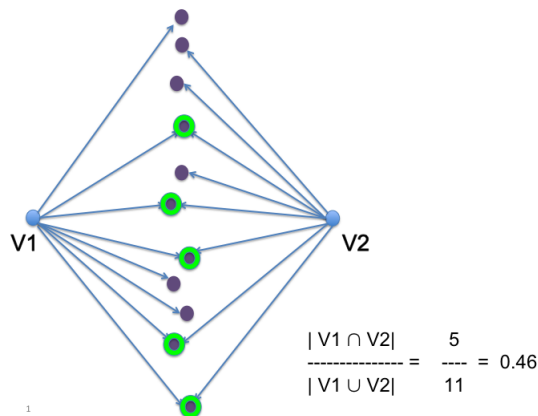


Fig. 3. Example of Jaccard Scoring

similar? Or, is the distribution of bytes per packet the same for systems in Europe as in North America?

This phase of the workflow is critical to an analytic process with meaningful, repeatable results. Performing it requires tools, methods, and (most significantly for time-sensitive analysis) time at the beginning of the analytic process.

B. Techniques

1) *Jaccard scoring*: Jaccard similarity for comparing two entities is defined as the cardinality of the intersection of values of the two entities divided by the cardinality of the union of their values. An example is shown in Fig. 3, where V1 and V2 each represent the flow data from one hour, with the circles between representing the eight most commonly visited Internet domain names, showing five entities in the intersection from a total of eleven entities in the union. The SME must judge whether that level of variability merits further investigation.

When analyzing Internet flow data, Jaccard scoring can be used to calculate the similarity between two time periods of the top 20 visited Internet domain names. The Jaccard calculation can be done either unweighted or weighted; *e.g.*, for the top-20-domains case, weighting would mean that the similarity between large number of visits to (say) google.com is weighted more than the small number of visits to another domain. Conversely, the weighting can be for rarity; *e.g.*, if we want to know whether two people are similar, the fact that they both visit a domain that is rarely visited population-wide means more than the fact that they both visit a very common domain.

2) *Synthetic discretization with Jaccard scoring*: The analytic applications discussed above are all examples where the analyst has subject matter expertise. Similarity scoring can also be used when the data is known but the meaning of the data is unknown; *i.e.*, to tell us if two data sets are similar enough that sense making can be skipped, thus semi-automating this critical step at the beginning of the analytic workflow.

Within a semantic context, we can apply Jaccard scoring to the predicate and object types found in a data set. This

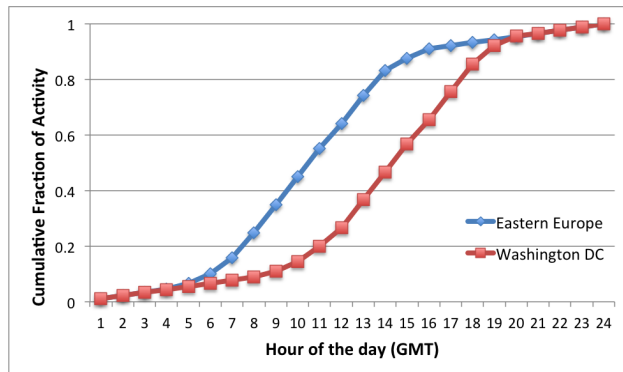


Fig. 4. Example of Kolmogorov-Smirnov Test

requires two extensions of the canonical algorithm in section IV.A below. These extensions correct for very popular nodes and handle different types of nodes (categorical similarity).

3) *Kolmogorov-Smirnov*: Once similarity scoring demonstrates a high degree of similarity in the predicate- and object-type composition of two data sets, another approach that can be applied is non-parametric goodness-of-fit statistical comparison between degree distributions of subjects with identical semantic types, using tests such as Kolmogorov-Smirnov [7] or Mann-Whitney U. For example, for each semantic type, the degree distribution for each predicate type is generated for the old, analytically-proven dataset and the new, believed-similar dataset. These degree distributions are compared via non-parametric goodness-of-fit statistical tests. The success or failure for each semantic-type/predicate-type pair is noted and then the aggregate success and failure counts are presented to the analyst at the end of the evaluation.

In Fig. 4, we use Kolmogorov-Smirnov to compare two sets of flow data, revealing that the time (measured from GMT) of the activity is distinctly different, with the blue data reflecting the normal work day of eastern Europe and the red data reflecting that of the US East coast.

4) *Graph Algorithms*: In many cases further insight can be gained from semantic data by directly analyzing it as a graph. For instance, knowing a set of IP addresses and the volume of data between each pair of addresses, the betweenness centrality algorithm will calculate which IP addresses are most *central*, giving insight into which might be most important for covert communication, or most disruptive were they to be disabled. Other graph metrics that may be valuable are the length of the shortest paths between entities, the communities that emerge by analyzing the connectivity of the graph, and the most likely path between entities (vertices). Such analysis of social networks is well known, but any transactional network lends itself to similar analysis. In our work we have used community detection [10], betweenness centrality [3], and BadRank [4].

IV. IMPLEMENTATION WITH RDF/SPARQL

Much of our work to date has focused on the RDF data representation [9] and SPARQL query language [11] implemented on Cray's Urika-GDTM graph appliance.

A. Jaccard scoring

Both the intersection and union steps in Jaccard scoring map trivially to SPARQL constructs. In the source listing below, an almost-identical subquery is repeated 4 times, once for calculating the cardinality of each set in the intersection and once for calculating the cardinality of each set in the union. The first instance of that subquery is in lines 9-14. It focuses on the first set of data (L11) residing in a named graph `g1`, defined by the `PREFIX` statement (L2), and specifically the data denoted by relationship (edge-type) `:myPred`. Those instances are grouped by the subject, counted within each group, ordered by descending count, and then only the 20 highest-count subjects are retained. Those partial results are joined with the same from the second set by the enclosing query (L7-21), which counts the number of distinct resulting subjects. The other subquery (L22-37) similarly calculates the union, with the sole substantive difference (besides variable names) being the `UNION` keyword inserted (L30) between the two subsubqueries to denote that results that appear in either subsubquery should be retained. Finally the outer query (L5-38) uses the numerator and denominator values to calculate the final Jaccard score.

```
1 PREFIX : <http://mydata.org>
2 PREFIX g1: <urn:g/2014-04-03T10:00:00>
3 PREFIX g2: <urn:g/2014-04-03T12:00:00>
4
5 SELECT (?num/?denom AS ?jaccard)
6 WHERE {
7   { SELECT (COUNT(DISTINCT ?s) AS ?num)
8     WHERE {
9       { SELECT ?s (COUNT(?s) AS ?s1Ct)
10        WHERE {
11          GRAPH g1: { ?s :myPred ?o1 }
12        } GROUP BY ?s ORDER BY DESC(?s1Ct)
13          LIMIT 20
14        }
15      { SELECT ?s (COUNT(?s) AS ?s2Ct)
16        WHERE {
17          GRAPH g2: { ?s :myPred ?o2 }
18        } GROUP BY ?s ORDER BY DESC(?s2Ct)
19          LIMIT 20
20        }
21      }
22    { SELECT (COUNT(DISTINCT ?s) AS ?denom)
23      WHERE {
24        { SELECT ?s (COUNT(?s) AS ?s1Ct)
25          WHERE {
26            GRAPH g1: { ?s :myPred ?o1 }
27          } GROUP BY ?s ORDER BY DESC(?s1Ct)
28            LIMIT 20
29          }
30        UNION
31        { SELECT ?s (COUNT(?s) AS ?s2Ct)
32          WHERE {
33            GRAPH g2: { ?s :myPred ?o2 }
34          } GROUP BY ?s ORDER BY DESC(?s2Ct)
35            LIMIT 20
36          }
37        }
38    }
```

B. Graph Algorithms

Graph algorithms can be executed via SPARQL in two ways. The first is to implement the algorithm in SPARQL, often by a succession of queries [12], with the benefit of having great flexibility to implement the algorithm and the downside of needing to implement the algorithm one's self. The second, for a small set of algorithms on Urika-GDTM, is to call a *built-in graph function* (BGF) in Summer 2014 or later releases [8], with the benefit of ease of execution and optimized performance, for the supported functions. The query below illustrates a simple call to the community-detection function. The `INVOKE` function calls the designated BGF and the `PRODUCING` keyword assigns the results to SPARQL variables.

```
1 CONSTRUCT {
2   ?v1 ?count ?v2
3 } WHERE {
4   SELECT ?v1 ?v2 (COUNT(?v2) as ?count)
5   WHERE {
6     ?v1 :myPred ?v2
7   } GROUP BY ?v1 ?v2
8 } INVOKE yda:community()
9 PRODUCING ?v ?communityID
```

V. CONCLUSION

We have used an analytic approach successfully with several different customers requiring discovery in highly diverse Big Data. The approach grows from initially no knowledge of the data to eventual deep knowledge, by enabling an analyst to interact directly with the data and apply her domain knowledge and intuition. We have implemented this approach with RDF and SPARQL on Cray's Urika-GDTM graph-analytic appliance. We believe this approach represents one way of addressing the critical challenge of quickly discovering deep knowledge in highly diverse Big Data.

REFERENCES

- [1] S. Al-Saffar et al, *Structure Discovery in Large Semantic Graphs using Extant Ontological Scaling and Descriptive Semantics*, Int'l Conferences on Web Intelligence and Intelligent Agent Technology, IEEE, 2011.
- [2] M.J. Franklin, *Making Sense of Big Data with the Berkeley Data Analytics Stack*, Proceedings of the 25th International Conference on Scientific and Statistical Database Management. ACM, 2013.
- [3] L. C. Freeman, *A set of measures of centrality based on betweenness*, Sociometry, 40, 35-41, 1977.
- [4] T.G. Kolda et al. *Generalized badrank with graduated trust*. Technical Report SAND2009-6670, Sandia National Labs, Albuquerque, NM, 2009.
- [5] T. Mattson et al. *Standards for graph algorithm primitives*. In High Performance Extreme Computing Conference, 2013 IEEE (pp. 1-2).
- [6] P. Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et des Jura*. Impr. Corbaz, 1901.
- [7] F.J. Massey, *The Kolmogorov-Smirnov test for goodness of fit*. Journal of the American Statistical Association, 1951.
- [8] D. Mizell et al, *Extending SPARQL with Graph Functions*, High Performance Big Graph Data Management, Analysis, and Mining, IEEE, 2014.
- [9] RDF Working Group, *Resource Description Framework (RDF)*, 2004.
- [10] E. J. Riedy et al, *Parallel Community Detection for Massive Graphs*, Lecture Notes in Computer Science, Volume 7203, pp 286-296, 2012.
- [11] S. Harris et al, *SPARQL 1.1 Query Language (W3C Recommendation 21 March 2013)*, <http://www.w3.org/TR/sparql11-query/>, 2013.
- [12] R. Tschentlin et al, *Implementing Iterative Algorithms with SPARQL*, Third International Workshop on Querying Graph Structured Data, 2014.