ICT Seventh Framework Programme (ICT FP7)

Grant Agreement No: 318497

Data Intensive Techniques to Boost the Real – Time Performance of Global Agricultural Data Infrastructures

SemaGrow

D4.3. RDF Triple Generator of Realistic Data Sets

Deliverable Form	
Project Reference No.	ICT FP7 318497
Deliverable No.	D4.3
Relevant Workpackage:	WP4: Rigorous Experimental Testing
Nature:	R
Dissemination Level:	PU
Document version:	Final V1.0
Date:	31/10/2014
Authors:	UAH, NCSR-D
Document description:	This document describes the implementation of the *RDF Triple Generator*. In particular it describes the simulation algorithm in which the *RDF Triple Generator* is based as well as the overall architecture and its components

Document History

Version	Date	Author (Partner)	Remarks
Draft v0.1	03/09/2014	NCSR-D	Document Setup
Draft v0.5	15/09/2014	NCSR-D	Draft Version
Draft v0.7	22/09/2014	UAH	Internal Review
Version 1.0	31/10/2014	NCSR-D	Delivered as D1.4.1

EXECUTIVE SUMMARY

This document describes the implementation of the *RDF Triple Generator*, the technical component able to generate data sets that project the growth and properties of the real ones in given time periods, which will be used in T4.3: Data Growth Trends & Projection. In particular it describes the architecture and the components of the *RDF Triple Generator* as well as the simulation algorithm responsible for the generated data sets.

.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Purpose and Scope

This document describes the implementation of the *RDF Triple Generator*, the technical component able to generate data sets that project the growth and properties of the real ones in given time periods, which will be used in T4.3: Data Growth Trends & Projection. In particular it describes the architecture and the components of the *RDF Triple Generator* as well as the simulation algorithm responsible for the generated data sets.

1.2 Audience

This deliverable has been created specifically for the SemaGrow Partners, describing the *RDF Triple Generator* the technical components which will be used in T4.3: Data Growth Trends & Projection.

1.3 Deliverable Structure

The document is structured in the following manner:

Chapter 2: **RDF Triple Generator Architecture:** This chapter presents and describes in detail the architecture and the components of the implemented RDF Triple Generator.

Chapter 3: **Monte Carlo Simulation:** This chapter describes in brief the basic Monte Carlo Simulation methods along with their positive and negative aspects.

2. RDF Triple Generator Architecture

2.1 Description

Realistic Data Generator (RDG) is a general purpose RDF triple generator. The generated data, produced by a combination of Monte Carlo methods, follow similar data distributions to a set of given real-life datasets. RDG architecture enables the parallelization and distribution of data generation process, supporting multiple TripleStore sources that follow different schemas/ vocabularies

2.2 Architecture & Components

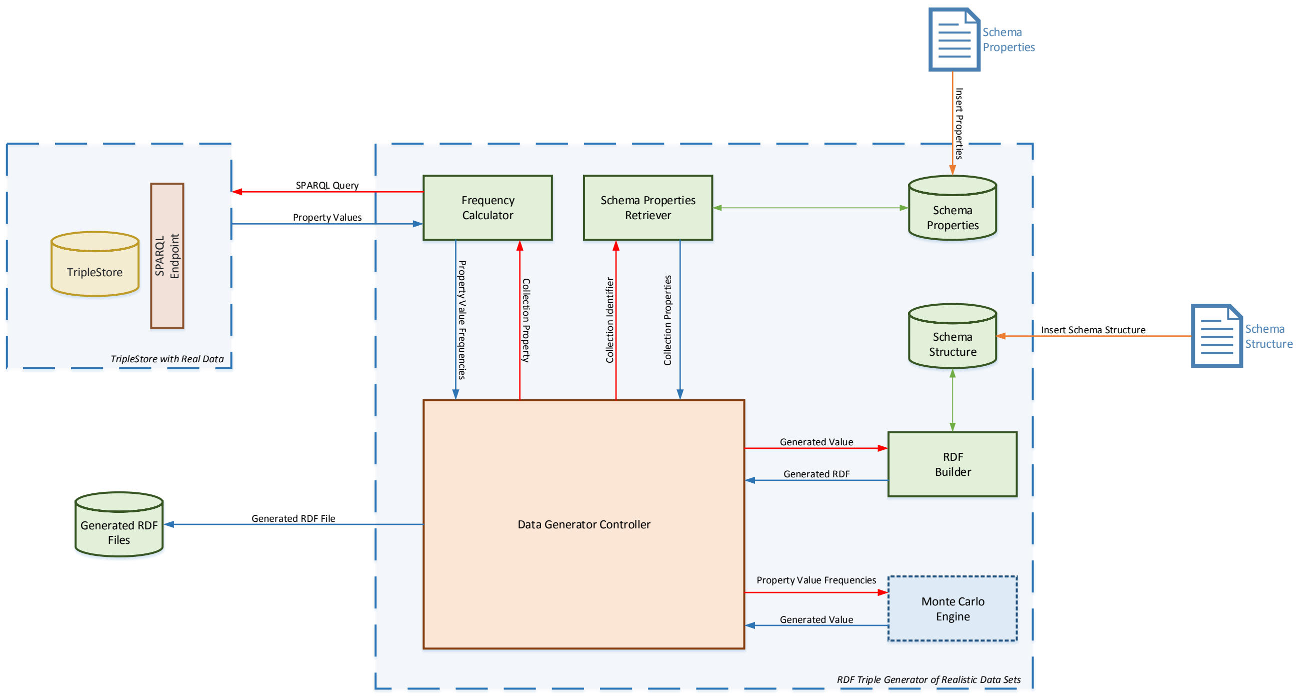

Figure 1: RDF Triple Generator Architecture

1. **Schema Properties Retriever (SPR)**

   SPR, provided a Collection Identifier, loads the appropriate vocabulary properties of the certain collection. The Collection Identifier is a unique identifier which is associated with a Vocabulary Identifier. Thus SPR can retrieve the correct properties from the Schema Properties Database (SPD). SPD should contain all the properties of a vocabulary. Also in the case of property that has discrete and defined range of values, those values should be also inserted in the SPD. Without the SPD the RED Triple Generator cannot function, thus the configuration and the initialization of the database is a basic prerequisite.

Table 1: Example of LAFLOR Vocabulary stored in Schema Properties Database (SPD)

Example of LAFLOR properties	
Property Name	**Property Value**
laflor:structure	Atomic
	Collection
	Networked
	Hierarchical
	Linear

laflor:aggregationLevel	1
	2
	3
	4

2. **Frequency Calculator (FC)**

   FC is responsible to communicate with the different TripleStores that contain real data, in order to retrieve and normalize the frequencies of the values of a given property. FC takes as input a property of a collection, builds the SPARQL Query, connects to a TripleStore or a collection of TripleStores and retrieves the values of the property. Afterwards, normalizes the frequencies of the values in order to be used by the Monte Carlo Engine.

Table 2: A Query Example that Frequency Calculator builds. The fields' propertyName and value are filled with the corresponding value retrieved by the Schema Properties Retriever

SPARQL Query Example
PREFIX laflor: <http://www.semagrow.eu/schemas/laflor#> SELECT ?o (COUNT(?s) as ?sCount) WHERE { ?s propertyName ?o . ?s propertyName value }GROUP BY ?o

3. **Monte Carlo Engine (MCE)**

   The Monte Carlo Engine implements the Acceptance-Rejection method in order to generate data that are realistic. Acceptance-Rejection method was chosen because fits better than other methods to the requirements of the task. Inverse Transform method was rejected because it is impossible to infer the Cumulative Density Function from the provided real data and Markov Chain Monte Carlo was rejected because the generated values are highly correlated which is hardly the case in the task at hand. MCE takes as input the frequencies of the values of a property. Using these values MCE estimates the probability density function (PDF) and afterwards encloses the (PDF) with a simple uniform distribution. Following the Acceptance-Rejection algorithm MCE generates values that are governed by the distribution of the observed data. The Acceptance-Rejection algorithm will be described in Chapter 2 of the current document.

4. **RDF Builder**

   RDF Builder is responsible to retrieve from the Schema Structure Database (SSD) the tags that are associated with the given property and to attach the generated value. Promptly, the tag with the generated value it attached to the tags generated before. SSD should be configured and initialized before the initiation of the *RDF Triple Generator*. More specifically, SSD should contain the star tag and the end tag of each property of a collection.

Start & End Tag Example		
Property Name	**Start Tag**	**End Tag**
laflor:structure	<laflor:structure>	</laflor:structure>
laflor:aggregationLevel	<laflor:aggregationLevel>	</laflor:aggregationLevel>
laflor:status	<laflor:status>	</laflor:status>

5. **Data Generator Controller (DGC)**

   DGC is the main component of the *RDF Triple Generator*. It is responsible for coordinating and controlling all the other components of the *RDF Triple Generator*. The first step of DGC in order to initiate the generation of realistic RDF Data Sets, is to load the properties of a schema using the Schema Properties Retriever. Afterwards, enables the Frequency Calculator in order to retrieve the value frequencies of the loaded properties from a Triple Store which contains real data. When the Frequency Calculator returns the calculated frequencies, DGC feeds the Monte Carlo Engine and receives from it a generated value. The generated value is redirected to the RDF Builder, where the RDF Builder loads the tags of the corresponding property and builds property by property the RDF structure containing also the generated values. Afterwards returns the result back and DGC stores the result as an RDF file to the file system.

3. Monte Carlo Engine Algorithms

3.1 Introduction

Monte Carlo Simulation is a category of computational algorithms and methods, which offer a practical way to sample random variables that are governed by complex or unknown probability density function [1][2]. Thus Monte Carlo Simulation is applied in a wide variety of filed such as physics, biology and mathematics. The fact that Monte Carlo Simulation can be used to model an unknown probability function inspired the application of these algorithms and methods to create generators that produce datasets that resemble real data.

3.2 Monte Carlo methods

3.2.1 Inverse Transform

Inverse transform method is the most basic Monte Carlo method [3]. Under the assumption that the density function $f(x)$ ranges on$-\infty < x < \infty$, its cumulative distribution function is given by

$$F(x) = \int_{-\infty}^{a} f(x)dx$$

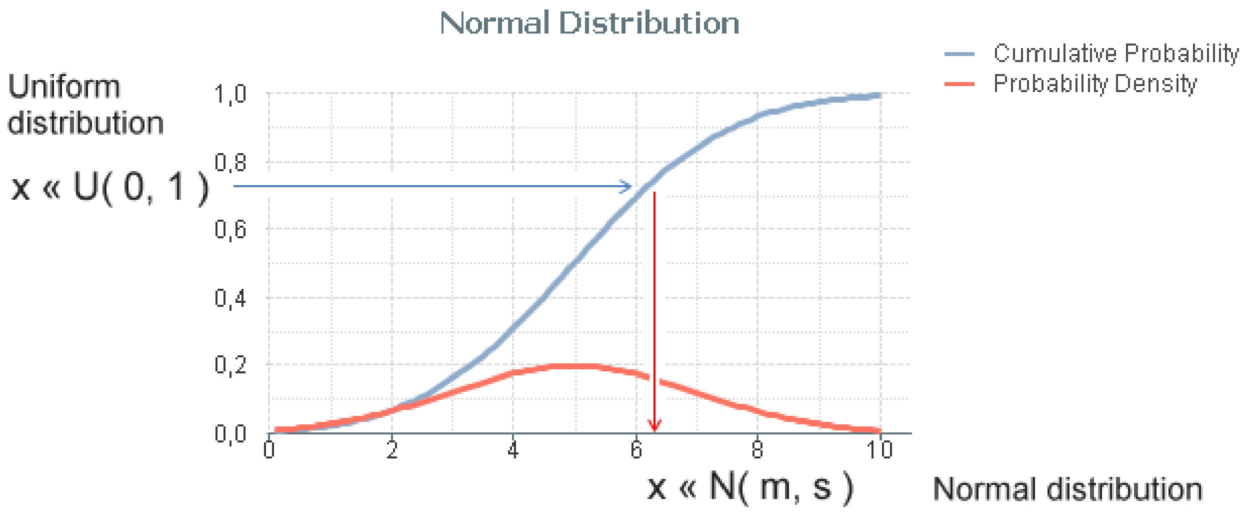

Figure 2: Sampling a random number from a uniform distribution (0,1) and finding a random number governed by the Normal distribution by using F(x)

Given a, the probability density is $f(a)$ and the integrated probability is $F(a)$. By assigning in a variable u a random value sampled from a uniform distribution with range $(0,1)$ and by setting $u = F(x)$ then (provided that the inverse of F can be found):

$$x = F^{-}(u)$$

x is a sample governed by the probability density function $f(x)$

Inverse Transform is a straightforward and clear approach, however rarely can be used in real world scenarios, because most of the times the analytic form of the Cumulative Probability is unknown or it is too complex to find the inverse.

3.2.2 Acceptance-rejection (Von Neumann)

Under the assumption that the probability density function $f(x)$ for a given value x can be computed then the Acceptance-rejection method can be used. More specifically, $f(x)$ is enclosed in a shape by a known and easily generated distribution $h(x)$ multiplied by a constant factor C. Thus it is possible to sample $h(x)$ and for every generated x to calculate $f(x)$. Any x that satisfies the restriction $uCh(x) \leq f(x)$ is accepted as a valid value governed by $f(x)$, if the restriction is not satisfied then the value is rejected and a new one is generated [4] [5]. Acceptance-rejection method, although its simplicity, is a powerful one since any probability distribution can be simulated with almost any prior knowledge. The drawback of this approach is that in complex distributions it will generate valid values in a slow pace.

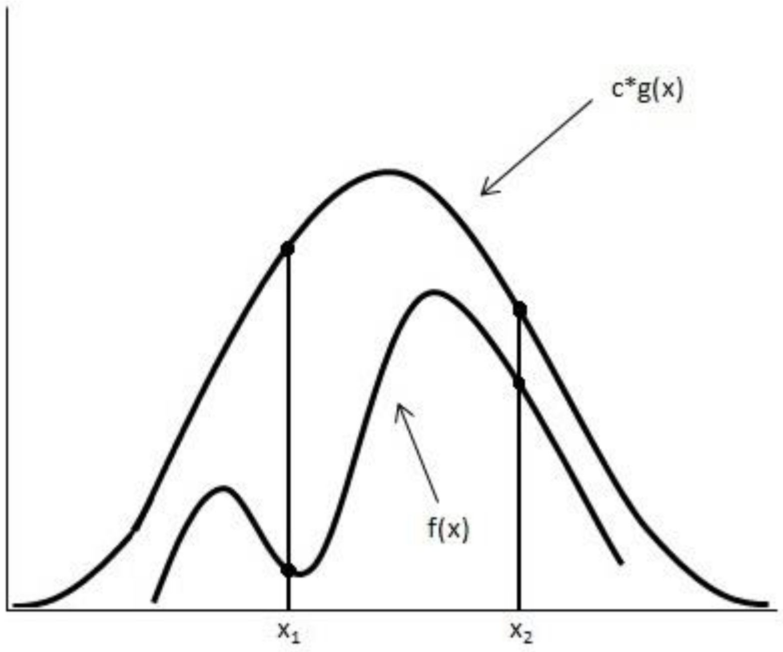

Figure 3: g(x) encloses f(x). Afterwards g(x) is sampled and the generated values are checked if satisfy the restriction

3.2.3 Markov Chain Monte Carlo

Another popular approach is the Markov Chain Monte Carlo (MCMC). Especially in cases where is impossible to use the Inverse Transform method and the Acceptance-Rejection method is impractical to use [6] [7]. In MCMC we assume a probability density function $p(\theta)$, where θ is a vector of parameters $\theta = (\theta_1, \ldots, \theta_n)$. MCMC samples from $p(\theta)$ and calculates the marginal distribution. A widely used algorithm which implements MCMC is the Hastings algorithm. This algorithm generates multidimensional θ points, using a target probability density function proportional to $p(\theta)$. More specifically, assuming a multivariate Gaussian which is centred in θ_0.the algorithm steps are

1. Generate a value θ using the assuming probability density function $q(\theta; \theta_0)$.
2. Calculate the Hasting ration, $a = \min\left[1, \frac{p(\theta)q(\theta;\theta_0)}{p(\theta_0)q(\theta;\theta_0)}\right]$
3. Generate a uniform distributed value u in $[1,0]$
4. If $u \leq a$, $\theta_1 = \theta$ otherwise $\theta_1 = \theta_0$
5. Set $\theta_0 = \theta_1$ and repeat from step 1

The difficulty on MCMC lies on the fact that the steps needed to converge cannot be estimated and more sophisticated MCMC algorithms add a large cost of additional computation time. However the most important drawback of this method is that once the algorithm converges the generated values will be highly correlated, thus the samples will be biased which is not the case in many real-life scenarios.

4. References

[1] Metropolis, N. and Ulam, S. "The Monte Carlo Method." *J. Amer. Stat. Assoc.* **44**, 335-341, 1949

[2] Metropolis, N. "The Beginning of the Monte Carlo Method." *Los Alamos Science,* No. 15, p. 125.

[3] L. Devroye, Non-Uniform Random Variate Generation (Springer-Verlag, New York,1986)

[4] J. von Neumann, "Various techniques used in connection with random digits. Monte Carlo methods", Nat. Bureau Standards, 12 (1951), pp. 36–38.

[5] Robert, C.P. and Casella, G. "Monte Carlo Statistical Methods" (second edition). New York: Springer-Verlag, 2004

[6] Rubinstein, R.Y.; Kroese, D.P. (2007). Simulation and the Monte Carlo Method (2nd ed.). Wiley. ISBN 978-0-470-17794-5

[7] Gilks, W.R.; Richardson, S.; Spiegelhalter, D.J. (1996). Markov Chain Monte Carlo in Practice. Chapman and Hall/CRC.