TOntoGen: A Synthetic Data Set Generator for Semantic Web Applications

Matthew Perry
Large Scale Distributed Information Systems (LSDIS) Lab
Computer Science Department, The University of Georgia, Athens GA 30602-7404
{mperry}@cs.uga.edu

The development of algorithms and applications for the Semantic Web requires high-quality ontologies for testing and validation. Our work on the SemDis [6] project centers on discovering complex relationships between entities. This requires the development of path discovery algorithms for RDF graphs, and we have found that it is often difficult to find sufficient data sets for testing these algorithms. Often publicly available, populated ontologies such as TAP [7] and SWETO [1] are lacking in either richness of relationships or sufficient instantiation of defined relationships. SWETO is our first attempt at creating a test bed for evaluating scalability and performance of Semantic Web techniques, technologies and tools. SWETO aims to create a large, populated ontology spanning many domains which uses real-world data extracted from various high-quality web sources. Using real-world data has several benefits for testing applications. However, the characteristics of the resulting knowledgebase of the populated ontology depend on availability of data rather than the actual real-world characteristics of the domain. At the same time, depending on real-world data is also limiting in terms of the types of test suites that can be created for evaluating various scalability and performance aspects of Semantic Web related algorithms, techniques and tools.

Realizing these limitations, we have developed TOntoGen (Test Ontology Generator), a synthetic data set generation tool which allows precise control over the ontology *population* process. We expect this to complement testbed ontologies based on real-world data (such as SWETO). The tool has been designed as a Protégé [5] plugin. After creating an ontology schema in Protégé, the tool allows specification of parameters controlling the relative distributions of instances of each class and property type in addition to the total number of class and property instances desired. The tool then generates an RDF instance graph with the specified characteristics. If the ontology schema is based on the real world domain semantics, then TOntoGen can be seen as a way to simulate population of that ontology.

Because ontologies are intended to model the real-world, the instance data of these ontologies should also match real-world characteristics. The parameters we use for graph generation are intended to capture these characteristics. For example, in an ontology modeling the college domain, it is unlikely that a college with 20,000 students will have only 10 professors. It would be more realistic for the college to have 1,500 to 2,000 professors. The central idea of specifying parameters is to indicate the relative numbers of instances of each class and property type. This ensures that the generated instance data accurately matches the expected instance data of the ontology. Parameters are set for the total number of nodes n_{nodes} and the total number of edges n_{edges}. Let C be the set of classes in the ontology and P be the set of property types. Each class $c_i \in C$ has an associated non-negative integer parameter α_{c_i}. The probability of generating an instance of class c_i is given by:

$$pr_{c_i} = \frac{\alpha_{c_i}}{\sum_{c_j \in C} \alpha_{c_j}}$$

Similarly, for each property type $p_i \in P$ a non-negative integer α_{p_i} is specified, and the probability of generating an instance of property p_i is given by:

$$pr_{p_i} = \frac{\alpha_{p_i}}{\sum_{p_j \in P} \alpha_{p_j}}$$

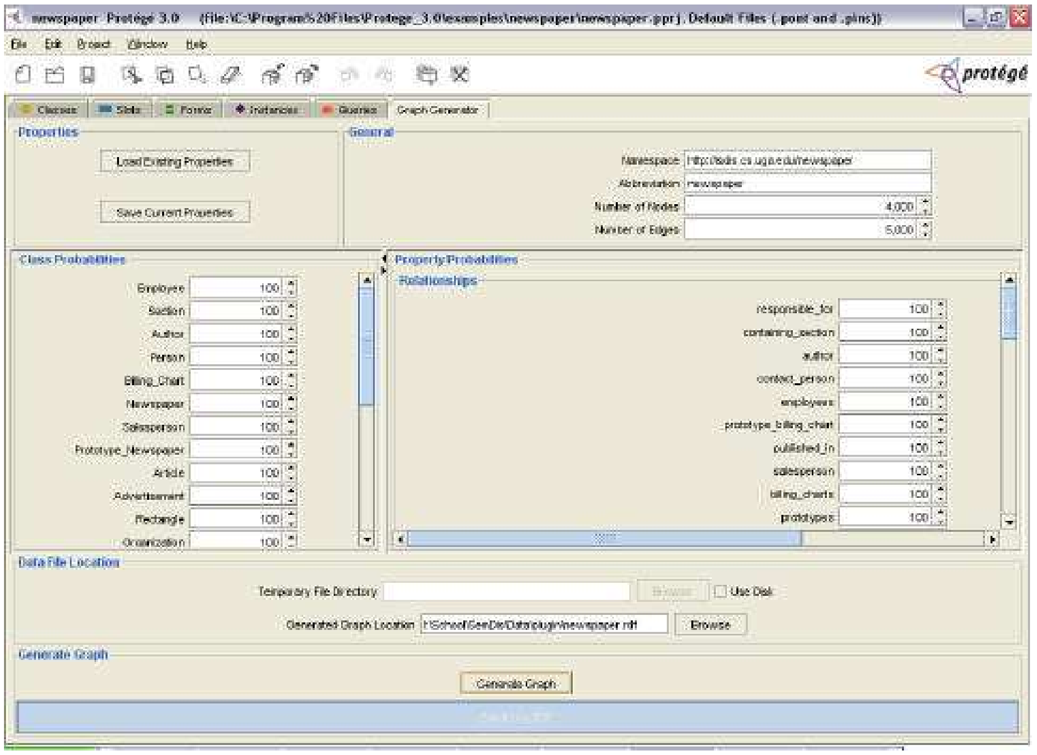
When generating an instance of a property, an instance of the domain or range is chosen as follows. First a class type must be chosen from the set of allowed classes (the domain or range class and its subclasses). Let D be this set of classes. The probability of selecting a class $c_i \in D$ is given by:

$$pr_{c_i,D} = \frac{\alpha_{c_i}}{\sum_{c_j \in D} \alpha_{c_j}}$$

After choosing this class an instance is chosen with uniform probability. For generating literal-valued properties, the generation of the property can be *probabilistic* or *exact*. The probability of generating a *probabilistic* literal property is computed as discussed previously. For an *exact* literal property, exactly one instance is created for each instance of each domain class, and the corresponding value of α_{p_i} is 0. A text file containing allowed values can be specified for each literal property. Values will be selected randomly from this file when creating an instance of the literal property.

TOntoGen provides two options for memory management during graph generation. For small and medium sized graphs, the tool holds the entire graph in memory during its creation, allowing graphs to be generated in seconds. For very large graphs, Java™ serialization is used to page parts of the graph into and out of memory during its creation. This limits the memory bottleneck when generating large graphs.

So far, TOntoGen has been used for testing two applications in SemDis [3, 4]. The test ontology used was a medium sized graph of 30,000 class instances and 45,000 property instances. These applications needed a test set which would allow for searching Semantic Associations [2] spanning multiple domains. To create this test set, three separate ontologies were created: one for the sports domain, one for the business domain, and one for the entertainment domain. For each ontology, TOntoGen created a 10,000 node and 15,000 edge instance graph. To allow for Semantic Associations spanning multiple ontology schemas, we had to create multi-classified instance nodes (for example, an entity which is classified as both an actor in the entertainment ontology and a spokesperson in the business ontology). We used a separate post processing application to merge entities from different domains based on a user-specified merging probability for pairs of class types. We plan to add this merging functionality to TOntoGen in the near future. Figure 1 shows a screenshot of the TOntoGen plugin. This tool can be downloaded from http://lsdis.cs.uga.edu/projects/semdis/tontogen/.



Namespace: http://lsdis.cs.uga.edu/newspaper
Abbreviation: newspaper
Number of Nodes: 4,000
Number of Edges: 5,000

Employee	100
Section	100
Author	100
Person	100
Billing_Chart	100
Newspaper	100
Salesperson	100
Prototype_Newspaper	100
Article	100
Advertisement	100
Rectangle	100
Organization	100

responsible_for	100
containing_section	100
author	100
contact_person	100
employees	100
prototype_billing_chart	100
published_in	100
salesperson	100
billing_charts	100
prototypes	100

Temporary File Directory: ☐ Use Disk
Generated Graph Location: I:\School\SemDis\Data\plugin\newspaper.rdf

Figure 1. Screenshot

References:

1. Boanerges Aleman-Meza, Chris Halaschek, Amit Sheth, I. Budak Arpinar, Gowtham Sannapareddy, SWETO: Large-Scale Semantic Web Test-bed, The 16th International Conference on Software Engineering & Knowledge Enginerring (SEKE2004): Workshop on Ontology in Action, Banff, Canada, June 21-24, 2004, pp. 490-493. Also http://lsdis.cs.uga.edu/projects/semdis/sweto/
2. Kemafor Anyanwu, Amit P. Sheth. ?-Queries: Enabling Querying for Semantic Associations on the Semantic Web, The Twelfth International World Wide Web Conference, Budapest, Hungary (2003).
3. Maciej Janik, Krzysztof J. Kochut. BRAHMS: A Workbench RDF Store And High Performance Memory System for Semantic Association Discovery. University of Georgia Technical Report 05-002.
4. William H. Milnor, Cartic Ramakrishnan, Matthew Perry, Amit P. Sheth, John A. Miller, Krzysztof J. Kochut. Discovering Informative Subgraphs in RDF Graphs. University of Georgia Technical Report 05-001.
5. Protégé. http://protege.stanford.edu/.
6. Semantic Discovery: Discovering Complex Relationships in the Semantic Web. http://lsdis.cs.uga.edu/projects/semdis/.
7. TAP. http://tap.stanford.edu/.

Acknowledgements: I thank all SemDis project members especially Cartic Ramakrishnan for his helpful comments and suggestions during formulation of graph generation parameters. This material is based upon work supported by the National Science Foundation under Grant No. IIS-0325464 titled "SemDis: Discovering Complex Relationships in Semantic Web". Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.