VIVO Harvester

Introduction

The VIVO Harvester is a library of tools designed to read and transform data from external data sources and ingest it into VIVO or potentially any other triplestore or semantic platform. The library was originally developed by the University of Florida Harvester Team during the 2009-2011 NIH Grant.

The VIVO Harvester is currently maintained on GitHub by John Fereira from Cornell as part of VIVO-related projects including AgriVIVO and USDA VIVO. Other contributions to ongoing Harvester enhancements have been made by Alex Viggio through Symplectic, Ltd.

Source

- Recommended Harvester branch to check out or download from Git

Architecture and flow

The VIVO Harvester is a collection of small Java tools which are meant to be strung together in various ways to create a harvest process that is custom-tailored to your needs and (importantly) repeatable. This architecture makes the Harvester extremely versatile, but at the same time presents a bit of a learning curve.

We highly recommend that you become familiar with the basics of semantic technologies including RDF and ontologies and download and install the VIVO software before embarking on a data ingest process. Try entering sample data ranging from people and their affiliations to publications, grants, or awards and honors; then export the RDF from VIVO to see what it looks like – for many people having an example is much more intuitive than interpreting ontology diagrams or writing RDF directly.

This following vignettes attempt to follow the steps of a "typical" harvest with a focus primarily on functionality, not configuration or execution.

Fetch

The first step of a typical harvest is the get you data from your target source. We call this the Fetch. For example, let us suppose we have a VIVO installation containing researchers at our university, and we want to harvest from Pubmed information on publications written by researchers at our university. In this case we would use Harvester's PubmedFetch tool to send a query off to Pubmed, which will return the results of that query to us in its own XML format. The harvesters Fetch package (org.vivoweb.harvester.fetch) contains various methods for retrieving data from external data sources, including as CSV files, as JSON, through JDBC calls, or via OAI Harvest.

Translate

The next step of a typical harvest is the translation. The fetched data will be in its own format, and this needs to be converted into VIVO-compatible triples. If the input is an XML format, this can be done using the XSLTranslator tool and a .xsl file containing XSLT code specific to the data format being converted to RDF/XML triples. Included with Harvester in the config/datamaps/ directory are several pre-written XSLT files for frequently-needed formats (including for example Pubmed). Another standard method in harvesting data is to prepare a SPARQL Construct using the VIVO UI that will take in RDF data and transform it into the VIVO ontology. You can use the SPARQL Translator to process SPARQL Construct files against target models.

Score

Depending on your data the next step may be to match incoming data with data already in VIVO. For example, if you have just pulled in some publication information from Pubmed, you might want to compare the author names with people in your VIVO, so that you can link the publications with the authors. This comparison is done via the Score tool, which compares any values you want between VIVO and the input data, and assigns a number to the comparison.

Match

The Match tool will look at the numbers generated by Score and compare them to a threshold value. Input entities compared by Score that meet or exceed the threshold will have their identities changed to the URI of the person in VIVO, so that when the data is finally pulled into VIVO the new data will be linked to existing data. In this way you can fetch publications for your existing researchers.

Change Namespace

Depending on how your data came in and how you generated triples for it the last step before importing the information into VIVO is to give your data proper URIs via the ChangeNamespace tool. Prior to this step, URIs may be placeholders provided by the XSLT translation (typically using aspects of the raw data that are expected to be unique, such as an ISBN number) or blank nodes from a SPARQL Construct. If you've generated unique URI's for all of your data using a piece of unique information then you can skip this step. After this step all data has a proper VIVO URI and is ready for import into VIVO.

Update

This step allows for multiple Harvester runs in succession to recognize data that has been modified since the previous run and update accordingly. A "previous harvest model" is created, which on the first run contains all the data imported on that run. On subsequent runs, this is compared with the new data to determine triples that have been removed or added since the last run. This comparison is made by the Diff tool, and the output is an "Additions file" and a "Subtractions file", containing RDF/XML data that should be added and removed, respectively, from VIVO.

Transfer

The data from the Additions file is added both to VIVO and the previous harvest model in two separate calls of the Transfer tool. Then the data from the Subtractions file is removed both from VIVO and the previous harvest model in two more Transfer calls.

At this point a harvest is complete.

Next Steps

Included in Harvester's scripts/ directory are several sample scripts which have been tested and will perform different types of harvests. One of the best ways to get started is to find one that is close to your needs, test it on a test server or virtual machine, and then tweak it until it meets your needs.

Read the Harvester User Guide to learn more about using the harvester.

- Typical harvest

- University of Florida Harvester Team
- University of Florida Harvester Documentation Archive

Harvester Instructions

Harvester User Guide

Scheduling

Walkthrus for example scripts

JDBC Example Script

Pubmed Example Script

IP Example Script

Deployment

- Execute
- Scheduling
- Configuration

Video Walkthrus

Screencasts of example harvester runs: https://sourceforge.net/projects/vivo/files/VIVO%20Harvester/Demonstration/

Case Studies/Examples

University of Florida PubMed Harvest

University of Florida PeopleSoft Harvest

University of Florida Department of Sponsored Research Harvest

Class Documentation

- Fetch
- Translate
    - GlozeTranslator
    - SPARQLTranslator
    - XSLTranslator
- Score/Match

    - Score
    - Algorithm
    - Match
- Transfer
- Diff
- Qualify
    - ChangeNamespace
    - RenameResources
    - Smush
- Utilities
    - JenaConnect
    - RecordHandler
    - ArgParser
    - Merge
    - DatabaseClone
    - XPathTool
    - XMLGrep
    - CSVtoJDBC

Development Documentation

- Harvester Development Toolkit
- Harvester Documentation Procedures
- Harvester Planned Features
- Advanced PubMed name matching diagram
- New Harvest Workflow Proposal
    - RecordHandler Tool Specification
    - RecordCompare Tool Specification
    - SPARQLFetch Tool Specification
    - Update Tool Specification
- Arbitrary Sparql Structure Scoring

Milestones

- Milestone 1
- Milestone 2
- Milestone 3
- Milestone 4
- Milestone 5
- Milestone 6
- Milestone 7
- Milestone 8
- Mileston