Goals and Approach

Goals

Ability to search across multiple VIVO installations. This means:

- harvesting information from several independent installations of VIVO or other software that can produce RDF compatible with the VIVO ontology in one of 3 (or possibly more) ways
    - responding to linked open data requests in one of several RDF serializations
        - note that this may be directly from a VIVO application or from Harvard Profiles
        - or from another application configured to return RDF
            - e.g., Iowa's Loki software does not store data natively in RDF but can return it in response to linked data requests
            - or using D2R (http://d24q.org)
            - or using tools such as John Fereira's semantic services, although these were designed to deliver data from VIVO to other applications not configured to consume RDF directly
    - returning an entire file of RDF from a web-accessible directory (a file with only the statements about the URI requested; it my also be possible to return one big file containing that URI)
    - responding to SPARQL query requests from a public SPARQL endpoint
        - or, if the harvesting tool is provided with credentials, from a private SPARQL endpoint
- indexing the information harvested, including the original URI in the source system and a subset of the content associated with that URI in the source system, to facilitate text-based searching
- providing a simple, Google-like search with options to limit in advance by type of result (e.g., people, organizations, publications, events)
- providing results that have been relevance ranked across the sources being searched, in contrast to federated searches
- providing short snippets of text for each result to aid interpretation
- providing faceted display to aid users in filtering results; the two current facets are source institution and the type of result
- linking back from each result to the source so that the full scope of the result can be seen in its original context

Search

1. What features are desired for the search?
2. What type of search?
3. What is the goal of the search?
    a. Full text? - *yes*
    b. "semantic"? - *future – the indexing takes advantage of the semantic structure of the VIVO ontology to include relevant text in the Solr document for each entry, but the search interface does not support queries that depend directly on the semantic relationships (e.g., find all principal investigators of grants investigating cancer who have collaborations with researchers on depression)*
    c. faceted? - *yes, though this could benefit from expansion*
    d. Complex queries? - *future*
    e. For people? - *yes*
    f. For publications, organizations, etc? - *yes, but needs further refinement*

Approaches

Make a index to support the desired types of search and have a web site that facilitates user with querying that index. Keep that index up-to-date.

Approach to building the index

1. For each institution
    a. Get a list of all URIs of interest for that institution
2. For each URI
    a. Get the linked data RDF for the URI
    b. Build a Solr index "document" using the RDF statements for which that URI is the subject; subsequent request obtain additional data for related objects based on VIVO's linked data harvesting patterns, that will add to the index a person's title(s) from their position(s) and other data from their VIVO page (real or virtual, if from another system) that would normally be indexed with that person in VIVO's internal search
        - *TODO: what governs the follow-on linked data requests, and do the results from what is harvested into a local VIVO search?*
    c. Add the document to the Solr index

Notes

- Same as what is currently in place
- Current VIVO does not have a direct way to get institutional URIs; VIVO has the option of differentiating internal from external URIs for any type of entity, and this could be useful in harvesting only institutional URIs pertaining to the source of the system.
- VIVO used to get RDF for each URI, then make subsequent requests as needed
  - Can investigate new approaches
- Policy questions
  - How much data do we want to get from each resource (e.g. people)
  - This is the kind of thing that needs to be asked of the institutions
  - Suggestion to collect these tasks in a spreadsheet
    - Include time estimates, and outstanding questions
  - How to determine when external resources have changed

Approach to keeping the index up-to-date

1. For each institution
   a. Get a list of URIs that have been modified, based on the last modified date for that URI in the source system's internal VIVO search index
2. For each URI
   a. Calculate what individuals are affected by this modification
   b. Add to update list
3. For each URI in update list
   a. Get the linked data RDF for the URI
   b. Build a document using that data
   c. Add the document to the Solr index

Notes

- Hope is that the approach is same as building the index, with different input

Alternatives Approaches

TODO: what other approaches are there?

- what use should be made of the institutional internal class, if populated, to limit data harvested to what is part of the institution harvested (note that we can't rely on this being populated, especially for data not produce by the VIVO software
  - this may not always be the intended effect – e.g., it may be desirable to harvest funding agencies but not the names of the institutions listed with educational training
- should the data harvested align with what is included in a VIVO internal search, or be much more limited (both by harvesting only certain types and by doing fewer follow-on queries for data closely related to the individual being harvested

Technology Choices

1. There are some parts of the technology stack that are suggested by the goal of indexing data from VIVO.
   - Using HTTP requests for RDF to gather data from the sites is the most direct approach.
   - Most other options for gathering data from the VIVO sites would need additional coding.
2. In general we would go with Solr for the search index because of we have experience with it, because of its documentation, because of it distributed features and because it is mature.
3. As of 2012 vivosearch.org uses Drupal and solrsearch javascript libraries. The js libraries allow the development of the search UI with only client side interaction (https://github.com/evolvingweb/ajax-solr).
   - This choice could be revisited for the multi-site VIVO search project.
4. In order to scale the process out we were planing to use Hadoop to manage parallel tasks and to run the indexing jobs on a set VMs setup as Hadoop nodes.
   - Many approaches to the problem of indexing linked data from VIVO sites would be embarrassingly paralleled.
   - Brian Caruso Cornell has worked with RDF indexing to Solr on Hadoop clusters on Eucalyptus clouds.
   - Consider using a IaaS abstraction layer such as jclouds, apache libcloud or overmind. These allow developing against an interface which can then target many different cloud service providers. The primary goal of this would be to avoid lock in to one cloud provider.

Notes

- HTTP for retrieving RDF, yes
- What is the adoption of SPARQL in the community
- It may be nice to demonstrate that a SPARQL endpoint is not needed to enable interesting results
- Solr, seems reasonable for now
  - Considering having Solr in one place versus distributed Solr (master/slaves)

- Web interface: drupal with solrsearch.js
    - Most work is on clientside with js
    - This continues to be appealing
    - We have limited insight into this component
    - Suggestion to create list of default technologies, criteria, and alternatives
- Hadoop is currently reasonable choice
- Ruby (blacklight/hydra) or Drupal?
    - The js pattern allows from minimal reliance on Drupal
- Need a mock-up of the UI to inform design of solr index
- BootStrap is an interesting js framework to consider
- Drupal upgrade cycle can be onerous

Technology Alternatives

1. We could use a different index software other than Solr.
    - What would that be?
    - A database server with full text capabilities?
    - What are other options?
    - Are there full text search NoSQL options?
2. What the the alternatives to Hadoop?
    - What other ways would sufficient management of multiple tasks?
    - Could we just do it as multiple java processes or multiple java threads?
    - OSGi?
    - Some of the hadoop related systems like hadoop Streaming or Cascade?
3. Serving the web site could be done with just about any system that allows interaction with Solr.
    - The solrsearch javascript libraries would allow any system that serves HTML and js to server this.
    - The options are expansive: httpd, wordpress, movible type, drupal, cold fusion.
    - If the solrsearch javascript can provide almost all of the interactivity on the client side it might be desirable for the server side be as simple as possible.
    - It may even be possible to use static HTML and .js files served by any old web server.

Index Updates

1. Once an index was created how would it be updated?
    a. Rebuild the whole index?
    b. Get a list of modified individuals from each site and only reindex them?