building better search for bioinformatics

Lucene/Solr Workshop Day
February 2016

Charlie Hull - Managing Director
Tom Winch – Senior Developer
Matt Pearce – Senior Developer

charlie@flax.co.uk
www.flax.co.uk/blog
+44 (0) 8700 118334
Twitter: @FlaxSearch

BBSRC 20 Years of Pioneering Great British Bioscience

flax The Open Source Search Specialists

EMBL-EBI

BioSolr & PDBe - Introduction

- Protein Data Bank (PDBe)

- facet.contains (autosuggest)
  - https://issues.apache.org/jira/browse/SOLR-1387
  - In Solr 5.1

- **XJoin (augmenting SOLR search with external data)**
  - https://issues.apache.org/jira/browse/SOLR-7341

- Federated search

BioSolr & PDBe - Introduction

- Protein Data Bank (PDBe)

- facet.contains (autosuggest)
  - https://issues.apache.org/jira/browse/SOLR-1387
  - In Solr 5.1

- Xjoin (searching external sources)
  - https://issues.apache.org/jira/browse/SOLR-7341

- **Federated search**

Xjoin concepts

- You have data in an external data source which is not suitable for indexing in
- For example, data is from a live web service
- Match external data to augment your Solr search results for display or analys
  - and to influence the scoring and ordering of your results
- External data returned in a separate block in Solr results
  - similar to how highlights are returned

Xjoin implementation

- XJoin is implemented as a set of Solr plugin components
- One XJoinSearchComponent per external source
- External results match with Solr results via a “join id field”
- The join id field need not be the unique id field
- Java glue code needs to be written and installed on Solr server

Xjoin - bioinformatics applications

- In the PDBe context, external data is output of sequence similarity algorithms
- FASTA and PHMMER algorithms exposed as web services
- Glue code and config in BioSolr github

_

- Using pdb_id as the join id field
- Using similarity measures to boost score in Solr text search

Xjoin – other applications

- Filtering e-commerce search results, using price discount data
- Using clickthrough data to influence e-commerce search

Federated search - introduction

- Problem: we need to search across data sets split over multiple physical loca
- Data records may contain different fields, or the same fields but with differen
- Similar to pre-SolrCloud distributed search
- Records expected to appear in more than one data set, augmenting one anc

Federated search: Solr implementation

- Need to collect all documents with the same ID, not throw any away (don’t use groupin
- Merge documents into a unified schema
- Provide a query result count - statistical estimation
- How to score a document - incompatible scores from different servers

Federated search: challenges

- Solr ‘front end’ for SQL DBs, other search engines to enable federation
- Scoring is hard even within Solr
- Further funding applied for to develop federated search

BioSolr & SPOt – Indexing ontologies

Washington, N. & Lewis, S. (2008) Ontologies: Scientific Data Sharing Made Easy. Nature Education 1(3):5

Indexing ontologies – the problem

- You have a collection of documents annotated with ontology references.
- You want to search both the documents and the associated ontology data.
  - This may include associated nodes – “has location”, “is part of”, etc.
- Faceting the ontology references would be nice!
  - (especially if the facets can be presented in a tree)

Approach 1

lata separate

Approach 1 - steps

- Index the documents with the node annotations but no further detail.
- Index the ontology in its own core.
- Search the documents, then cross-match against the ontology.
- **BUT** requires multiple calls, doesn't allow searching both cores at the sam

Approach 2

data to your documents

Approach 2 – step 1

- Index the documents with the node annotations.
- While indexing, look up the node references, with their labels and synonyn
- Easier to include the ontology references in your search.
- Can boost fields as required.
- Faster to search

Approach 2 – step 2

- Expand the ontology data being stored.
  - Include single (or multi) level parent and child nodes, with their labels.

- Use dynamic fields to store additional relationships.

- Dynamic fields allow searches across specific relationship types (“is part o
- **BUT** requires additional Solr look-ups to be fully dynamic
  - (using /admin/luke to look through the current schema for dynamic fields).

Approach 2 – search screen

- General search box
- Options to include child and parent labels (one step removed)
- Dynamically-generated additional relationship search options

Approach 2 – search results

Approach 2 – final result

- We have developed a Solr UpdateProcessor to do this as part of the update ch
- The user defines the ontology location (file, URL) and the field to reference.
- Aims for convention over configuration for remaining properties.
  - Field names and ontology annotation details all customisable.

- *A similar plugin for ElasticSearch has also been developed - we'll talk about this tomorrow!;*

Aside: facet trees

- Additional search component to return facets from ontology references in tree f
- Extends FacetComponent
- Takes initial facets from results, searches hierarchical references to build the tr
- Avoids multiple calls to Solr from client-side to build the tree.

- A second collection can be used to search the ontologies...
  - **... BUT** it must be part of the same Solr instance

Aside: facet trees - challenges

- Nodes with multiple parents may appear more than or
  - … not yet found a solution for this!
- Default behaviour is to return entire tree.
  - Not always useful – may need to drill through several
- Solution: prune the tree!

Aside: facet trees - pruning

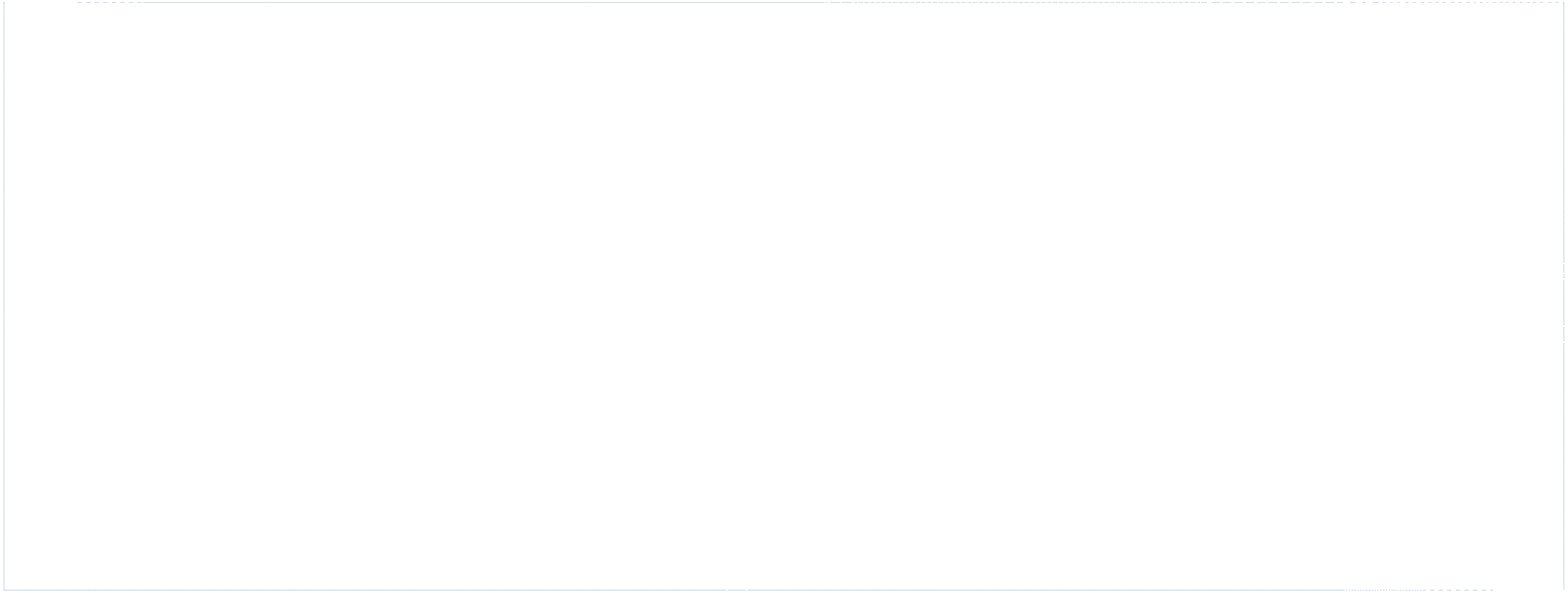
- Multiple pruning options available:
  - Simple pruning: remove layers with no useful informa
  - Datapoint pruning: return the highest counts at top-le

Approach 3

- Search the ontology, and cross-match with the documents.

- Allow SPARQL queries over the ontology index.
  - Enables complex searches over ontology relationships.
  - (SPARQL is a semantic query language)

Approach 3



Adding Apache Jena

- We use Apache Jena to provide TDB-querying with SPARQL.

  - Jena uses Solr to search specified text fields – set via configuration files.
  - Uses its own Triple Store for other fields and relationship queries.
  - Return the reference URI in the returned fields to cross-match with documents
  - Use a filter query to choose the matched documents.
  - Not that different from Xjoin!

Generic ontology indexer

- Stand-alone application to index different ontologies
- Allows separate configuration for each ontology
- Plugins may be used:
  - At item level, to external data some/all nodes;
  - At ontology level, to push the ontology into MongoDB, etc.

- Cross-pollination with the EBI's Ontology Lookup Service site (currently in beta

Get Involved!

- Check out the github page: https://github.com/flaxsearch/BioSolr
- Vote for Xjoin: https://issues.apache.org/jira/browse/SOLR-7341
- Suggest more use cases for what we've built!

Thank you for listening – any questions?

.co.uk/blog

700 118334

@flaxsearch