bel Documentation

Release 0.5.6-dev

William Hayes, David Chen

Dec 18, 2019

Contents:

Welcome to the BEL Resource Tools documentation.

CHAPTER 1

Informational Badges

1.1 Quickstart

1. *git clone git@github.com:belbio/bel_resources.git* to clone the BEL Resources project locally
2. Make sure BEL.bio API is running with the elasticsearch and arangodb containers - Instructions
3. *make install* will setup a virtualenv and pip install the requirements
4. Setup your configuration file (*Configuration*)
5. *make load_all* will download remote namespace/orthology files, convert into load-able formats and then load into elasticsearch and arangodb (*Namespaces*)
6. *make clean_all* will clean out all loaded namespaces and orthologies

Remote database files (e.g. Entrez Gene, SwissProt) are downloaded to <repo_dir>/downloads (gzipped). BEL Namespace files for loading into elasticsearch are created from source database files and stored in <repo_dir>/data/namespaces. Orthology files are stored in <repo_dir>/data/orthologs.

1.2 Overview

Work in progress

1.3 Configuration

Configuration is described in the BEL Python package documentation

Configuration documentation

1.4 Namespaces

1.4.1 Overview

This document describes how the Namespaces are processed. Most Gene/RNA/Protein Namespaces also include orthology equivalences.

The namespaces are collected from their original databases, e.g. the ChEBI database. They are then converted into terminology and orthology JSON files as described in the *BELBio Schema Repository <https://github.com/belbio/schemas/tree/master/schemas>*.

The namespace individual entries are stored in Elasticsearch to provide strong, scalable, and fast search capabilities. The namespace equivalents and orthologies if available are stored in ArangoDB for graph queries.

1.4.2 Processing Overview

1. Run the Namespace script (for download and source file processing) to generate each Namespace <prefix>.jsonl.gz files in *tools/namespaces*
   (a) First downloads original Namespace database files such as Entrez Gene and compresses them using gzip if not already gzipped.
      i. Only if it is newer than the prior download if determinable (some FTP servers don't make filedates available)
      ii. If source file modification dates are indeterminable, it will download if the local file is older than 7 days (configurable in *belbio_conf.yml <http://bel.readthedocs.io/en/latest/configuration.html>*) - this is also adjustable per namespace script.
   (b) Namespace script then processes the original source files to create the term data, equivalence and hierarchy
   (c) Namespace script writes out each term into a gzipped JSONL file using the terminology schema
2. Load namespaces into Elasticsearch

1. After setting up Elasticsearch index
2. Run *tools/load/load_elasticsearch.py -a*
3. Load equivalence files

1. Generate equivalence files to load into ArangoDB from <term>.jsonl.gz files
2. Load equivalence files into ArangoDB using *tools/load/load_arango.py*

5. Run Orthology scripts in tools/orthologs
   1. First Download Original Orthology database files and compress using gzip
   1. If it is newer than prior download if determinable (some FTP servers don't make file modification dates available) (orthology and terminology source files use same download location so will only download once if file(s) is used by both terminology and orthology scripts)
   2. If source file modification dates are indeterminable, it will download if the local file is older than 7 days (configurable in *belbio_conf.yml <http://bel.readthedocs.io/en/latest/configuration.html>*) - this is also adjustable per namespace script.
   2. Orthology script then processes the original source files to create the orthologous relationships
   3. Orthology script writes out each orthology into a gzipped JSONL file using the orthologs schema
   4. Load orthology datasets into ArangoDB using *tools/load/load_arango.py*

1.4.3 Namespace Scripts

Each namespace script is an independent script. Most do utilize some utility functions from a utility library supporting the namespace and orthology scripts.

Note: Namespace values (NSArg) need to be quoted if they contain whitespace, comma or ')'. This is due to how BEL is parsed. An NSArg (namespace:term, e.g. namespace argument of a BEL function) is parsed by looking for an ALLCAPS namespace prefix, colon and then a term name. The parsing continues for the term name until we find a space, comma or end parenthesis ')'. If the term contains any of those characters, it has to be quoted using double-quotes.

Note: Any character except an un-escaped double quote can be in the NSArg if it is quoted including spaces, commas and ')'.

Generally the namespace scripts do two main things:

1. Download the namespace source datafiles
2. Build the <term>.jsonl.gz file

All of the namespace scripts will be stored in the resource_tools/namespace directory. Any *.py files in that directory will be run to (re-)create the <namespace>.jsonl.gz files. The namespace scripts will create the <namespace>.jsonl.gz files in the resource_tools/data/namespace directory. Any *.jsonl.gz files will be loaded into Elasticsearch into the namespace index.

1.4.4 Taxonomy Terminology

Taxonomy IDs are based on the NCBI Taxonomy. Taxonomy is treated just like other terminologies with additional features of taxonomy_name object and taxonomy_rank (kingdom, . . . , genus, species). **The Taxonomy terminology script has to be run first as it creates the taxonomy_labels.json.gz file which is used by all terminologies that stores species_id and species_label in the <term>.jsonl.gz files**.

The taxonomy_labels.json.gz file is a map (dictionary/hash) of all of the TAX:<int> versus labels but only for taxonomy entries with taxonomy_rank: "species". **Note: It may be necessary to add labels to this file for entries with non-species taxonomy_rank as several EntrezGene and SwissProt namespace entries do not have labels in this file.**

The Taxonomy Namespace prefix is 'TAX'. Humans have the taxonomy id of TAX:9606 with a custom label of 'human'.

Custom labels for specific species are sourced from the *taxonomy_labels.yaml* file adjacent to the taxonomy.py terminology script. Custom labels file looks like:

```
# Override taxonomy label
# taxonomy_src_id: label
---
9606: human
10090: mouse
10116: rat
7955: zebrafish
```

1.4.5 Orthology Scripts

Orthology Gene/Protein IDs collected from their source files need to be converted to the canonical Namespace for Genes/Proteins (currently Entrez Gene, prefix EG) prior to loading into ArangoDB **TODO**. This will save time in processing through the equivalence edges.

1.4.6 Terminology and Orthology Schemas

Schemas for terminologies and orthologies are kept in the BELBio Schema Repository.

1.4.7 Elasticsearch Index

The Elasticsearch index map is in the es_mapping_term.yaml file and the index is created using the setup_es.py script. This setup_es.py script must be run before loading the terminologies the first time. It will delete the *terms* index if it already exists. **Note: Need to setup an A/B index option so that we can switch the index alias to a new terms index.**

1.4.8 ArangoDB

A 'bel' database is created and the following collections are added and loaded:

1. ortholog_nodes
2. ortholog_edges
3. equivalence_nodes
4. equivalence_edges

These collections of nodes and edges allow equivalence and orthology queries to be run against the bel ArangoDB database.

1.5 Glossary

This is maintained in the BEL.bio API documentation

BELBio Glossary

1.6 Contributing

Important: First off, thanks for taking the time to contribute!

The following is a set of guidelines for contributing to BEL.bio and its repositories, which are hosted in BELbio on GitHub. These are mostly guidelines, not rules. Use your best judgment, and feel free to propose changes to this document in a pull request. If you have any questions, please let us know.

You will need to sign a Contributer's License Agreement (CLA) the first time you submit a pull request and anytime afterwards that the CLA agreement is updated (very rarely). This is signed by clicking on the request as part of the pull request and digitally signed with your Github ID automatically.

1.6.1 Code of Conduct

This project and everyone participating in it is governed by the *Code of Conduct* *Code of Conduct*. By participating, you are expected to uphold this code. Please report unacceptable behavior to one of the project maintainers, William Hayes or Anselmo Di Fabio.

1.6.2 Code Contributions

This project uses the Git Forking workflow as discussed here https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow

Tip: It may be helpful to review your plans with the community before starting work.

1. Please add an issue to the repository prior to working on a feature or bug.
2. Fork and clone the repository
3. Create a topic branch
4. After completing your changes, please ensure that the code style outlined in Editorconfig is followed.
5. Submit a pull request referencing the issue being resolved

- Other examples of Git Forking workflow http://www.asmeurer.com/git-workflow/ or http://blog.scottlowe.org/2015/01/27/using-fork-branch-git-workflow/

1.7 Code of Conduct

Contributor Covenant Code of Conduct

1.7.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

1.7.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances

- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

1.7.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

1.7.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

1.7.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at William Hayes or Anselmo Di Fabio. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

1.7.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

1.8 DevOps

All administrative tasks are managed by running make tasks using the top-level Makefile in the project folder.

1.8.1 Builds and Testing

We will use TravisCI for Open Source to run builds and tests.

1.8.2 Documentation

Documentation is based on Sphinx and created and hosted by ReadTheDocs using the *belbio* user id.

1.8.3 Dependabot

We use https://app.dependabot.com/accounts/belbio/repos to keep the python module requirements up to date. It uses the *belbio* user id.

1.8.4 Code Quality

We are using Code Climate for code quality assessments.

We are using CodeCov for code test coverage assessments.

1.8.5 Contributor Licensing Agreements

All pull requests require signing the [CLA Assistant](https://cla-assistant.io/) Contributor's License Agreement.

1.9 Changelog

CHAPTER 2

Related Documentation

- BEL API documentation
- BEL Python Package