Contents

1 LinkExplorer

The LinkExplorer is a web-based tool for exploring nodes and relations of link prediction benchmark datasets and explanations of predictions done with the rule-based approach SAFRAN. A running instance of this tool can be found at:

http://explore.ai-strategies.org

Included are three biomedical knowledge bases:

- OpenBioLink
- Hetionet
- Pheknowlator

and two general-domain benchmarks:

- YAGO3-10
- WN18RR

Paper preprint on bioRxiv • Peer reviewed paper in the journal Bioinformatics (for citations) • Supplementary data • Citation (bibTex)

1.1 Tutorial

A screenshot tutorial of the main functionalities can be found at https://openbiolink.github.io/Explorer/.

1.2 BYOD (Bring your own data)

1.2.1 Dataset metadata (RDF)

LinkExplorer retrieves metadata of benchmark datasets from RDF graphs, which provides labels, descriptions of nodes and relations in the dataset, as well as the edges of the dataset. You can extend LinkExplorer with your own dataset metadata graph by specifying the endpoint of your RDF graph after clicking

on the button 'Load custom endpoint' in the dataset loading screen (`/loader`). Further information about the schema of the RDF graphs can be found here.

If you want to make your RDF graph publicly available, please contact us!

1.2.2 Explanation files

You can load any explanation file produced with SAFRAN, by clicking on 'Load local explanation' and selecting the generated `.db` file. Metadata graphs of custom explanation files can be either 'None', a public graph (From the dropdown) or a custom graph (see here).

1.3 Host LinkExplorer yourself

1.3.1 Docker

The LinkExplorer application can be run with *Docker*. We divide our application into three containers:

- Client (The frontend of LinkExplorer)
- API/Server (Hosts sqlite-Databases storing explanations of predictions in Benchmarks)
- Blazegraph/RDF-Database (Database for metadata (Labels, Descriptions, ...) of entities in Benchmarks)

The LinkExplorer application is orchestrated through *Docker Compose* and can be run following these steps:

1. Install Docker
2. Clone repository
3. Run `docker-compose up --build` from the `Explorer` folder

That's it!

Now the LinkExplorer app is accessible via http://localhost:5000, while Blazegraph is accessible via http://localhost:9999. Explanation files (sqlite) and the `index.json` should be added to `/server/db`. The host of all SPARQL endpoints that are running in the docker blazegraph container should be `blazegraph`, f.e.

```
{
    "Dataset": [
        {
            "ID": "wn18rr",
            "Endpoint":
↪   "http://blazegraph:9999/blazegraph/namespace/wn18rr/sparql",
            "Name": "WN18RR",
            "Version":  "",
            "Description":  "WN18RR is a link prediction dataset
↪   created from WN18, ...",
```

```
            "Explanation": [
                {
                    "ID": "max",
          ...
```

1.3.2 Manual execution

1. Clone repository
2. Start **server** by running (Starts backend on port 3001)

```
cd server
npm install
npm run start
```

3. Build and host **client**

```
cd client
npm install
npm run build
```

and host the static build with a web server. All `/rpc` calls have to be proxied to the backend node server.

Example with https://github.com/lwsjs/local-web-server

The following command hosts LinkExplorer client at port 5000 and proxies all `/rpc` calls to the backend.

```
ws --port 5000 --directory build --spa index.html --rewrite '/rpc
↪  -> http://localhost:3001/rpc'
```

4. Host your own SPARQL Endpoint

Generally you can use whatever graph database you want that supports RDF*/SPARQL*. We included a blazegraph, which can be started by running

```
cd server/blazegraph
python start.py
```

Further information can be found at https://blazegraph.com.

Optional If you want to populate the blazegraph server with the datasets mentioned above run `python setup.py` with a running blazegraph instance.

5. Open http://localhost:5000

1.3.3 Data

The central file which stores names, endpoints, explanations, ... of benchmarks hosted on the server is `index.json` which sould be stored under `/server/db`. A template can be found here.

Explanation files generated by SAFRAN (.db) should be stored in `/server/db` and should adhere to the following naming convention: `{ID of dataset}_{ID of explanation}.db`. F.e. the explanation file for WN18RR and Maximum aggregation should be `wn18rr_max.db` if the entry in the `index.json` is as follows

```
{
    "Dataset": [
        {
            "ID": "wn18rr",
            "Endpoint":
↪   "http://explore.ai-strategies.org:9999/blazegraph/namespace/wn
↪   18rr/sparql",
            "Name": "WN18RR",
            "Version":  "",
            "Description":  "WN18RR is a link prediction dataset
↪   created from WN18, ...",
            "Explanation": [
              {
                "ID": "max",
                "Label": "MAX",
                "Date": 1624529144,
                "Comment": "These results were retrieved by
↪   applying the MaxPlus (AnyBURL default) aggregation ...",
                "Method": "max",
                "RuleConfig": "SNAPSHOTS_AT = 1000 ...",
                "ClusteringConfig": ""
              },
        ...
```

1.4 Citation

Ott, S., Barbosa-Silva, A., & Samwald, M. (02 2022). LinkExplorer: Predicting, explaining and exploring links in large biomedical knowledge graphs. Bioinformatics. doi:10.1093/bioinformatics/btac068