OntoConnect with Graph Neural Network (pytorch - biggraph)

- Input:
  - source.json (Human)
  - target.json (Mouse)
- Output:
  - alignment

Model Info:

1. PyTorch-BigGraph (PBG) is released by Facebook's research team
2. Graph embedding methods learn a vector representation of each node in a graph by optimizing the objective that the embeddings for pairs of nodes with edges between them are closer together than pairs of nodes without a shared edge.
3. Graph embedding methods are a form of unsupervised learning, in that they learn representations of nodes using only the graph structure and no task-specific “labels” for nodes.
4. Working Principle
   1. List of edges

SOURCE	EDGE	DESTINATION
"http://human.owl#NCI_C41452"	"self"	"http://human.owl#NCI_C41452"
"http://human.owl#NCI_C41452"	"parent"	"http://human.owl#NCI_C22921"
"http://human.owl#NCI_C41452"	"child"	"http://human.owl#NCI_C13003"
"http://human.owl#NCI_C41452"	"child"	"http://human.owl#NCI_C19526"
"http://human.owl#NCI_C41452"	"eq"	"http://human.owl#NCI_C12928"
"http://human.owl#NCI_C41452"	"disjoint"	"http://human.owl#NCI_C41448"
"http://human.owl#NCI_C41452"	"restriction"	"http://human.owl#NCI_C41623"

   2. Initial Node Embedding (generated from FastText)

Node	100d/200d/300d vectors
http://human.owl#NCI_C41452	[, , , , ..]
...	...

3. The main idea of training the Graph embeddings
    1. The edges provided in dataset are considered as positive edges.
    2. It generates negative edges between the nodes which are not connected. These random “false” edges as negative training examples along with the true positive edges.
    3. Idea is maximize the score of positive edges and minimize the score of negative edges.

The **score function** is as follows
$f(\theta_s, \theta_r, \theta_d) = sim(g_s(\theta_s, \theta_r), g_d(\theta_d, \theta_r))$
sim is cos/dot/l2
$g_{(s/d)}$ is operator ~ none/translation/diagonal/linear/affine/complex-diagonal

The **loss function** is as follows
$\mathcal{L} = \sum_{e \in G} \sum_{e' \in S'_e} max(f(e) - f(e') + \lambda, 0)$
G id list of edges
S'_e set of negative edges for every positive edge
$f(e)$ score for a positive edge
$f(e')$ score for a negative edge
λ is regularization

Step-1 ~ ModifyLbl.ipynb

Modify Labels

- read the input files
    - /ip/source.json
    - /ip/target.json
- convert the "lbl" and populate "altLbl"
- save the output files
    - /modifylbl/source.json
    - /modifylbl/target.json

```
"http://human.owl#NCI_C41452": {
        "lbl": " Subependymal_Cell",
        "altLbl": "cell subependymal",
        "iri": "http://human.owl#NCI_C41452",
        "vector": null,
        "entityTyp": "Class",
        "parentCls": [
            "http://human.owl#NCI_C13003"
        ],
        "childCls": [],
        "eqCls": [],
        "disjointCls": [],
        "restriction": [
            "(<http://human.owl#UNDEFINED_part_of>,http://human.owl#NCI_C41448)",
            "(<http://human.owl#UNDEFINED_part_of>,http://human.owl#NCI_C41623)"
        ]
    }
```


Step-2 ~ CreateDictionary.ipynb

Create Dictionary

- read the input files
  - /modifylbl/source.json
  - /modifylbl/source.json
- create dictionary of all words present in the all entities.
  - No of Unique Words:- 2124
- save the output files
  - /dict/dict.txt

Step-3 ~ DictionaryToVector.ipynb

Dictionary To Vector

- read the dictionary file
    - /dict/dict.txt
- Get the FastText vector for each dictionary word.
- save the output file with vectors
    - /dict/dict.json

Step-4 ~ EntityToVector.ipynb

Entity To Vector

- read the dictionary file
    - /dict/dict.json
    - /modifylbl/source.json
    - /modifylbl/target.json
- copy the vectorsint the source and target file
- save the output file with vectors
    - /fastentity/source_fast.json
    - /fastentity/target_fast.json

Step-5 ~ GenWordSim.ipynb

Generate Word Similarity

- read the vectors for both the source and target
    - /fastentity/source_fast.json
    - /fastentity/target_fast.json
- create a json file that contains top-k similar human entities for each mouse entity
    - /output/word_sim/word_sim_cosine.json

```
{
    "http://mouse.owl#MA_0001080": {
        "http://human.owl#NCI_C32243": 0.20560630681970293,
        "http://human.owl#NCI_C33727": 0.20639166686063304,
        "http://human.owl#NCI_C33502": 0.21291418700858478,
        "http://human.owl#NCI_C32903": 0.2141350602468982,
        "http://human.owl#NCI_C12719": 0.2657356704378381
}
```


Step-6 ~ GenerateGraphData.ipynb

Generate Graph Data

- read the structures for both the source and target
  - /fastentity/source_fast.json
  - /fastentity/target_fast.json
- generate edges for for both source and target
  - /gnnentity/source_gnn.json
  - /gnnentity/target_gnn.json


```
"http://human.owl#NCI_C41452": {
"graphEdges": [
            ["http://human.owl#NCI_C41452", "self", "http://human.owl#NCI_C41452"],
            ["http://human.owl#NCI_C41452", "parent", "http://human.owl#NCI_C13003"],
            ["http://human.owl#NCI_C13003", "parent", "http://human.owl#NCI_C41452"]
            ["http://human.owl#NCI_C41452", "restriction", "http://human.owl#NCI_C41448"],
            ["http://human.owl#NCI_C41448", "restriction", "http://human.owl#NCI_C41452"],
            ["http://human.owl#NCI_C41452", "restriction","http://human.owl#NCI_C41623"],
            ["http://human.owl#NCI_C41623", "restriction","http://human.owl#NCI_C41452"]
        ]
}
```

Step-7 ~ EmbedOntoGraph.ipynb

Embed Onto Graph

- read both the source and target one by one
  - fastentity/source_fast.json
  - fastentity/target_fast.json
- Now for each entity in source/target file
  - extract the enitity name ~ NCI_C41452
  - get the updated embedding
    - first, it creates all the necessary files for the training
      - /gnnentity/entity_graph/src/[entity_nm]/graphs/
    - populate pre-embeddings (FastText) in the h5py file
    - train each graph (entity) with num_epochs (100)
    - retrieve the new embedding
  - store the new embedding it in dictionary for each entity
- store the dictionary in a new file, this will contain the new embedding for each entity both for source and target
  - /gnnentity/source_gnn_meta.json
  - /gnnentity/target_gnn_meta.json

Step-8 ~ GenMetaSim.ipynb

GenMetaSim

- read the vectors for both the source and target
  - /gnnentity/source_gnn_meta.json
  - /gnnentity/target_gnn_meta.json
- create a json file that contains top-k (same as word-sim) similar human entities for each mouse entity

  - /output/meta_sim/meta_sim_cosine.json

```
{
    "http://mouse.owl#MA_0001080": {
        "http://human.owl#NCI_C32243": 0.20560630681970293,
        "http://human.owl#NCI_C33727": 0.20639166686063304,
        "http://human.owl#NCI_C33502": 0.21291418700858478,
        "http://human.owl#NCI_C32903": 0.2141350602468982,
        "http://human.owl#NCI_C12719": 0.2657356704378381
    }
}
```

Step-9 ~ GenCombSim.ipynb

Generate Combine Similarity

- read the vectors for both the source and target
  - /output/word_sim/word_sim_cosine.json
  - /output/meta_sim/meta_sim_cosine.json
- create a json file that contains top-k similar human entities for each mouse entity
  - /output/output_final.json

```
[
   {
        "entity1": "http://mouse.owl#MA_0001087",
        "entity2": "http://human.owl#NCI_C12665",
        "measure": 1.0
   }
]
```

Step-10 ~ OntoEvaluation.ipynb

[Evaluation]

- read the vectors for both the source and target
  - /gold_copy/reference.xml
  - /output/output_final.json
- It prints the precision, recall and F-measure

```
Precision: 0.935
Recall: 0.710
F measure: 0.807
```

```
1
```

Resources

- **Main Paper: https://mlsys.org/Conferences/2019/doc/2019/71.pdf**
- https://torchbiggraph.readthedocs.io/en/stable/data_model.html
- https://torchbiggraph.readthedocs.io/en/latest/scoring.html#interpreting-the-scores
- https://torchbiggraph.readthedocs.io/en/latest/faq_troubleshooting.html
- https://github.com/facebookresearch/PyTorch-BigGraph
- https://github.com/facebookresearch/PyTorch-BigGraph/blob/master/docs/source/configuration_file.rst
- http://pages.cs.wisc.edu/~shivaram/cs744-fa20-slides/cs744-pytorch-biggraph-notes.pdf
- https://ai.facebook.com/blog/open-sourcing-pytorch-biggraph-for-faster-embeddings-of-extremely-large-graphs

CrtUtil.ipynb

[Create Utility]

- create constant (OntoSimConstants.py)

- create import (OntoSimImports.py)
- create parameter file (ontosim.json)
- create folders under data folder

Result Analysis

		100-dimension			200-dimension			300-dimension		
Number of Prediction	Similarity Threshold	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
Top-1	0.99	0.974	0.674	0.796	0.979	0.670	0.795	0.979	0.669	0.795
	0.98	0.935	0.710	**0.807**	0.967	0.692	0.806	0.973	0.683	0.803
	0.970	0.853	0.730	0.787	0.930	0.715	**0.808**	0.953	0.705	**0.810**
	0.96	0.751	0.752	0.751	0.880	0.733	0.800	0.916	0.721	0.807
	0.95	0.678	0.766	0.719	0.808	0.750	0.777	0.868	0.736	0.797
	0.94	0.619	0.775	0.688	0.746	0.764	0.755	0.815	0.750	0.781
	0.93	0.578	0.786	0.666	0.692	0.773	0.730	0.754	0.764	0.759
	0.92	0.553	0.792	0.651	0.648	0.786	0.711	0.713	0.774	0.742
	0.91	0.532	0.804	0.641	0.612	0.794	0.690	0.671	0.783	0.723
	0.90	0.512	0.810	0.628	0.581	0.801	0.673	0.641	0.789	0.708
Top-3	0.99	0.976	0.676	0.799	0.981	0.671	0.797	0.982	0.671	0.797
	0.98	0.940	0.714	**0.811**	0.969	0.693	0.808	0.976	0.685	0.805
	0.97	0.862	0.737	0.795	0.934	0.718	**0.812**	0.956	0.706	**0.812**
	0.96	0.764	0.765	0.766	0.885	0.737	0.805	0.920	0.724	0.810
	0.95	0.697	0.787	0.739	0.815	0.757	0.784	0.873	0.740	0.802
	0.94	0.637	0.799	0.709	0.758	0.776	0.767	0.821	0.756	0.787
	0.93	0.598	0.813	0.689	0.707	0.790	0.746	0.764	0.774	0.769
	0.92	0.574	0.822	0.676	0.665	0.807	0.729	0.725	0.787	0.755
	0.91	0.554	0.836	0.667	0.628	0.816	0.709	0.687	0.802	0.740
	0.90	0.534	0.844	0.654	0.598	0.825	0.694	0.658	0.811	0.726
Top-5	0.99	0.976	0.676	0.799	0.981	0.671	0.797	0.982	0.671	0.797
	0.98	0.940	0.714	**0.811**	0.969	0.693	0.808	0.976	0.685	0.805
	0.97	0.863	0.738	0.796	0.934	0.718	**0.812**	0.955	0.707	**0.812**
	0.96	0.766	0.767	0.767	0.885	0.737	0.805	0.920	0.724	0.810
	0.95	0.699	0.790	0.742	0.815	0.757	0.784	0.873	0.740	0.802

0.94	0.640	0.802	0.712	0.760	0.779	0.769	0.822	0.757	0.788
0.93	0.603	0.820	0.695	0.709	0.793	0.749	0.766	0.776	0.771
0.92	0.579	0.830	0.682	0.669	0.812	0.734	0.728	0.790	0.758
0.91	0.562	0.848	0.676	0.632	0.821	0.714	0.690	0.805	0.743
0.90	0.542	0.856	0.664	0.604	0.832	0.700	0.662	0.815	0.730