TP on Link Prediction

andrea.failla@phd.unipi.it

November 2024

`linkpred` is a python library designed to provide support to unsupervised link prediction analysis. `LinkPred` is developed and maintained by Raf Guns (University of Antwerp) You can install it using pip.

```
$pip install linkpred
```

This requires at least python 3.8. The library offers a wide number of unsupervised predictors organized into four families:

- *Neighborhood*: AdamicAdar, AssociationStrength, CommonNeighbours, Cosine, DegreeProduct, Jaccard, MaxOverlap, MinOverlap, NMeasure, Pearson,ResourceAllocation
- *Paths*: GraphDistance, Katz
- *Ranking*: SimRank, RootedPageRank
- *Miscellaneous*: Community, Copy, Random

The library is documented with docstrings (i.e., comments in code with examples), you can check it here: `https://github.com/rafguns/linkpred/tree/main`.

1 Unsupervised link prediction

1. Download data about co-acting relations during season 6 of Game of Thrones

   ```
   $wget https://andreafailla.github.io/uploads/data/got-s6-edges.csv
   ```

2. Explore the network's basic properties: number of nodes, edges, density, and clustering coefficient.
3. Select at least 3 methods and apply them to your dataset:

```
from linkpred import predictors
predictor = predictors.CommonNeighbours(g, excluded=g.edges())
results = predictor.predict()
```

4. Sort the predicted links by score and display the top 10 predicted edges. Do these predictions make sense based on the network's structure?

1.1 Evaluate Prediction Accuracy

1. We will test the predictors on data from the 7th season of GoT. You can download the data with

```
!wget https://andreafailla.github.io/uploads/data/got-s7-edges.csv
```

   Then, load the network in the same way as before. Store the network to a variable called `test`

2. Using the Pair object from linkpred, we compute the universe and test sets.

```
import random
import itertools
from linkpred.evaluation import Pair

# Exclude test network from learning phase
training = g.copy()
# Node set
nodes = list(g.nodes())
nodes.extend(list(test.nodes()))
# Compute the test set and the universe set
test = [Pair(i) for i in test.edges()]
universe = set([Pair(i) for i in itertools.product(nodes, nodes) if i[0]!=i[1]])
```

3. Apply the predictors to the training network

4. use the EvaluationSheet object (from linkpred.evaluation import EvaluationSheet) to evaluate the obtained prediction against the test

```
plt.plot(cn_evaluation.fallout(), cn_evaluation.recall(), label="Common Neighbors")
```

1.2 Community-Based Link Prediction

- Use the `cdlib` library to detect communities in the network (e.g., Louvain method). Install cdlib with pip first, in case you don't have it on your machine/Colab.

- Compare the accuracy of link prediction within communities versus across communities. Are links more likely to form within the same community?

2 Supervised Link Prediction

In this exercise, you will perform supervised link prediction using network properties and embeddings. Follow the steps below to complete the task.

1. Construct a feature set for each edge in the network. Some ideas: Link Prediction Scores, difference in centralities of the nodes, etc.

2. Label each edge as a positive sample (existing edge) or a negative sample (non-existing edge).

3. Split your dataset of labeled edges into a training set and a testing set. Make sure to balance the samples in each set, ensuring a representative mix of both positive and negative samples.

4. Using the training set, train a classifier. Feed the classifier with the computed edge properties as input features and the labels as target values. Adjust model parameters as needed.

5. Apply your trained model to the test set and predict whether each edge is likely to exist. Evaluate the model's performance