Link prediction

Hands-on Tutorial
Neo4J + PySpark

Raphaël Azorin - IASD FC

Program

28/06

Setup check

Introduction slides

Hands-on tutorial - Part 1: data exploration and preparation (notebooks 1-4)

30/06

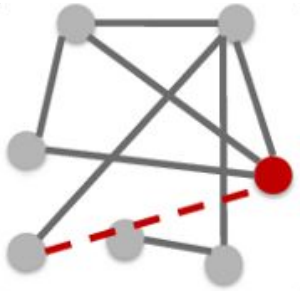
Hands-on tutorial - Part 2: modelling and evaluation (notebook 5)

Objectives

- Understand why link prediction is useful
- Get an overview of link prediction methods
- Identify real-world challenges
- Build a recommender system based on link prediction with a toy example

What is link prediction

- Predict a missing or future link in a network using its graph structure.
- Renewal of interest since 2004 with a paper from Liben-Nowell and Kleinberg

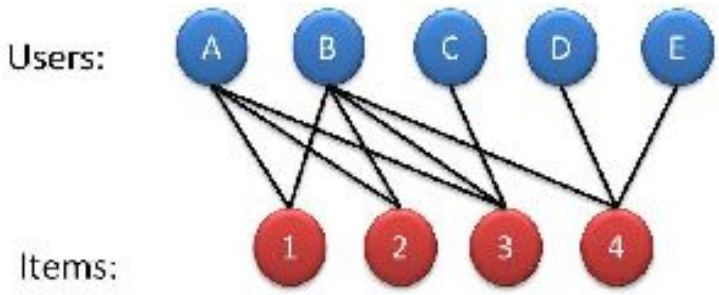


> Given a snapshot of a social network, can we infer which new interactions among its members are likely to occur in the near future? We formalize this question as the *link prediction problem*, and develop approaches to link prediction based on measures for analyzing the "proximity" of nodes in a network.

Use cases

Recommender systems

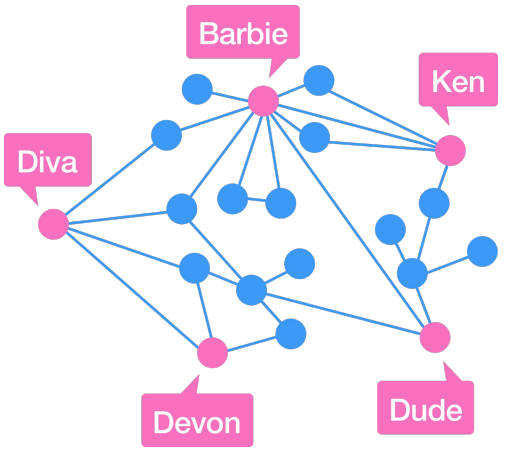
user-item graph



link prediction = recommending an item to a user

Networks analysis

social graph



link prediction = predicting a relationship

Use cases - Industries

Security
Counter-terrorism, mass monitoring

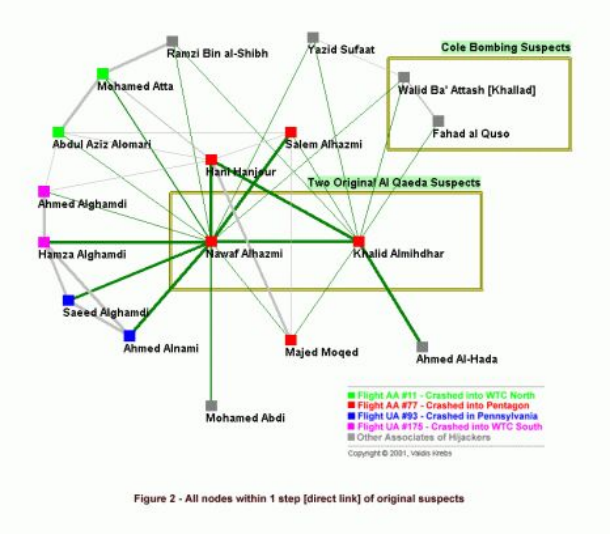
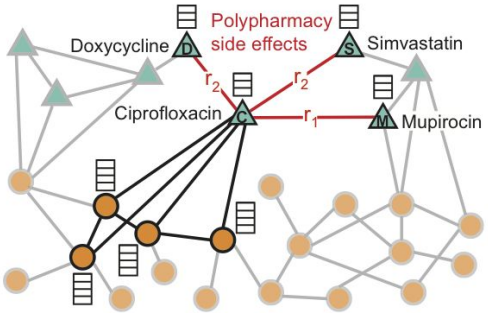


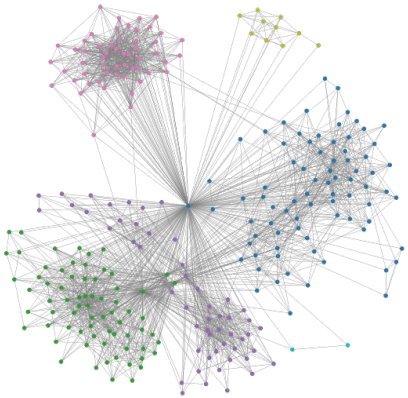
Figure 2 - All nodes within 1 step [direct link] of original suspects

Biology
Molecules network



△ Drug ● Protein
▭ Node feature vector
r_1 Gastrointestinal bleed side effect
r_2 Bradycardia side effect
△—● Drug-protein interaction
●—● Protein-protein interaction

Social media
People and communities



Methods - With heuristics

Idea

➔ Use a measure of proximity / similarity computed from the graph topology (number of common neighbors, Jaccard's coeff., etc.)

➔ If two unconnected nodes are close enough (according to a threshold), then they should be linked

Remarks

- Unsupervised method
- Which measure is the best one?

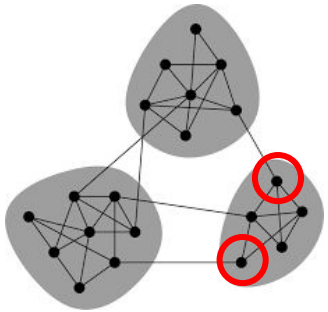
Indices	PPI	NS	Grid	PB	INT	USAir
CN	0.889	**0.933**	**0.590**	0.925	**0.559**	0.937
Salton	0.869	0.911	0.585	0.874	0.552	0.898
Jaccard	0.888	**0.933**	**0.590**	0.882	**0.559**	0.901
Sørensen	0.888	**0.933**	**0.590**	0.881	**0.559**	0.902
HPI	0.868	0.911	0.585	0.852	0.552	0.857
HDI	0.888	**0.933**	**0.590**	0.877	**0.559**	0.895
LHN1	0.866	0.911	0.585	0.772	0.552	0.758
PA	0.828	0.623	0.446	0.907	0.464	0.886
AA	0.888	0.932	**0.590**	0.922	**0.559**	0.925
RA	**0.890**	**0.933**	**0.590**	**0.931**	**0.559**	**0.955**

Comparison of similarity indices (AUC) - Lu 2010

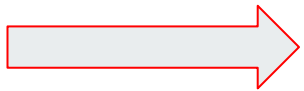
Methods - With supervised learning

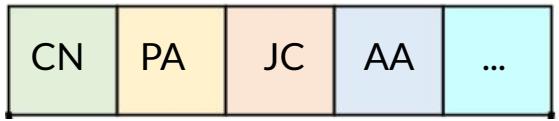
Idea

- ➔ Combine various similarity measure to predict if two nodes should be linked or not
- ➔ Train a binary classifier with features describing pairs of nodes (common neighbors, etc.)
- ➔ For two unconnected nodes, if the model predicts 1, then a link should exist between them, otherwise no.



Create a feature vector for a pair of nodes





Methods - With supervised learning

Process

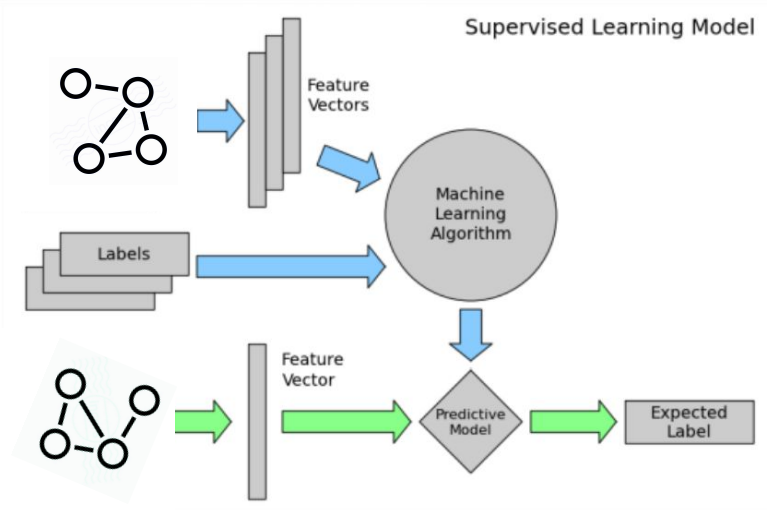
1. Extract pairs of nodes from the graph
   a. pairs of connected nodes (labeled with 1)
   b. pairs of unconnected nodes (labeled with 0)

2. Train / test split this dataset of pairs

3. Extract features for each pair (i.e. compute similarity measures)

4. Train the model (classification algorithm on tabular data)

5. Evaluate the model



Real-world considerations

Density and class imbalance

- ➔ Real graphs are usually very sparse, i.e. if we consider all the possible pairs of nodes in the graph, very few are actually linked.
- ➔ That is to say: the link density is very low (e.g. 0.00001).
- ➔ In the supervised learning approach, if we considered all possible pairs of nodes to build our dataset, we would have a lot more pairs labeled with 0 than with 1.
- ➔ We need to produce a train set with a 50% / 50% class repartition (ideal for learning). However, we should make sure that our test set respects the actual link density of our graph.

Class imbalance

Real-world considerations

Train and test sets design

➔ We should not share information between the train and test set to get a fair performance evaluation.

➔ The train/test split is challenging with graphs, because:

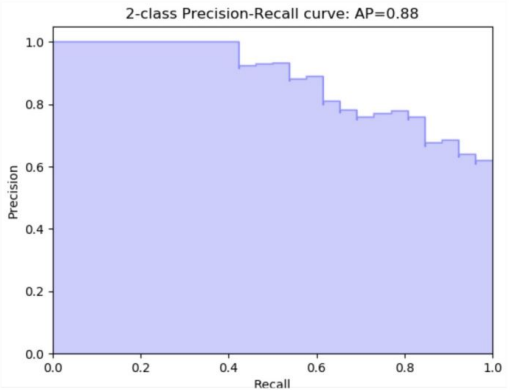
- it should not break any links
- each node should be in exactly on dataset (train or test)
- a node in the train dataset should not be linked to a node in the test dataset

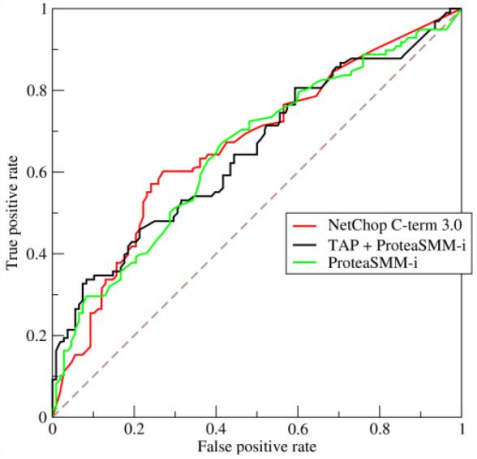
Real-world considerations

Model evaluation

➔ Just respecting the density in the test dataset is not enough because a “stupid” classifier that would always predict 0 still has a very high score and is very hard to beat.

➔ We can use *average* precision and recall: we compute these measures for various test datasets with various link densities and plot these results.





Recap

- Understand why link prediction is useful
  -> Various use cases, as long as data can be represented as a graph
- Get an overview of link prediction methods
  -> Unsupervised / supervised methods
- Identify real-world challenges
  -> Class imbalance and performance evaluation design

➔ Build a recommender system based on link prediction with a toy example

Download the notebooks from github.com/raphaaal/link-prediction

References

- *Link Prediction with GDSL and scikit-learn - Developer Guides*. Neo4j Graph Database Platform. (n.d.). https://neo4j.com/developer/graph-data-science/link-prediction/scikit-learn/.
- *Link Prediction - Developer Guides*. Neo4j Graph Database Platform. (n.d.). https://neo4j.com/developer/graph-data-science/link-prediction/.
- *ML Pipelines*. ML Pipelines - Spark 3.1.2 Documentation. (n.d.). https://spark.apache.org/docs/latest/ml-pipeline.html.
- *Use Apache Spark MLlib on Databricks*. Use Apache Spark MLlib on Databricks | Databricks on AWS. (n.d.). https://docs.databricks.com/applications/machine-learning/train-model/mllib/index.html#binary-classification-example.
- Cazabet, R. (n.d.). *Link Prediction and Graph Reconstruction*. CazabetRemy. http://cazabetremy.fr/Teaching/catedra/5-Linkprediction.pdf.