Universiteit Leiden

Master Computer Science

An empirical analysis of transfer learning for link prediction in real-world networks

Name: Emilio Sánchez Olivares
Student ID: s2985144
Date: 2023

Specialisation: Data Science

1st supervisor: Dr. Frank Takes
2nd supervisor: Dr. Akrati Saxena
3rd supervisor: Hanjo Boekhout MSc

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The task of predicting missing links in real-world networks has been thoroughly studied in the field of network science. Powerful algorithms have been developed that have shown remarkable performance in identifying unconnected pairs of nodes in a network likely to be connected. However, these algorithms often come at the cost of preprocessing and training time. Transfer learning has emerged as a viable solution to this problem, reusing previously trained models to predict on different data than they were trained on. However, limited research has been done on transfer learning for link prediction and how to efficiently and correctly employ a pre-trained link prediction model.

In this thesis we propose a transfer learning model for link prediction, with a particular focus on two important issues in this regard. The first is how to adequately test transfer learning between different train and test networks. The second objective is to get a detailed understanding of which networks perform well, and why. The latter is investigated by assessing the relationship between similarity in network structure and performance in terms of AUC scores. For this, we perform experiments on 49 real-world networks of varying size and domain in a supervised setting. We find that the transfer learning performance of a model is related to network statistics associated to how clustered the network used to train the model is. Additionally, we observed that transfer learning performance in general decreases when employed for networks with divergent topological properties.

Contents

1 Introduction **4**

2 Related Work **7**

3 Preliminaries **9**
3.1 Networks and Graphs 9
3.2 Network topology 9
3.3 Link prediction 11
3.4 Transfer learning 12

4 Methodology **13**
4.1 Data 13
4.2 Link prediction model 15
4.2.1 Pairwise training dataset 15
4.2.2 Features 16
4.2.3 Stacked classifier 18
4.3 Cross-validation training 18
4.4 Performance metrics 19

5 Experiments and results **22**
5.1 Experimental setup 22
5.2 Results 22
5.2.1 AUC scores matrix 22
5.2.2 AUC loss matrix 26
5.2.3 Tree-based algorithms for topology importances 28
5.2.4 Dissimilarities vs. AUC loss 30
5.2.5 Pairwise feature distributions case study 31

6 Conclusion **34**

References **35**

A Additional Material **39**

1 Introduction

The study of networks can be traced back to the beginning of graph theory proposed by Euler [15] and formalized by Biggs et al. [7]. These works describe a network as a set of discrete elements (nodes) linked by a set of connections (edges). Later, Erdős and Rényi in [14] noted that graph properties can be investigated in terms of probability distributions when they are regarded as stochastic objects. After growing interest in the study of networks, Barabási et al. [2, 3, 4] and Newman et al. [30] coined *network science*, which in contrast to previous work, focuses on the properties of real-world networks. This field, views networks as not static, but evolving over time; and aims to understand networks not only as topological objects, but as the framework upon which distributed dynamical systems are built.

Nodes and edges can be almost anything: scientific papers and citations, people and friendships (real or online), websites and hyperlinks. Additionally, networks can be classified in different ways. First, by their directionality, where edges that link two vertices asymmetrically are called directed, while edges link that two vertices symmetrically are called directed. Moreover, networks can also be classified by their modality. In bipartite networks there are two independent sets of nodes, and every set connects a node in one set to the other, whereas unipartite networks are only made up of a single set of nodes. Furthermore, real-world networks also differ from randomly generated networks in the sense that in random networks the average dominates, and the degree distribution (number of links per node) follows a Poisson distribution. On the other hand real-world networks, the degree distribution follows a power law distribution, where a few nodes contain the highest amount of links. Because of this, in real-world networks the averages are not meaningful, but instead, when networks grow, nodes prefer to attach to pre-existing hubs rather than to any node randomly. [5]

Networks can grow and change over time, with new connections happening as a network evolves. Liben-Nowell et al. [23] called this the *link prediction problem*, where the aim is to accurately predict the edges that will be added to a network during a certain time interval. Usually, this is achieved by training a supervised model on existing links and leveraging the proximity of node pairs. Some unsupervised learning approaches have also been suggested [21, 28]. Since then, link prediction has drawn attention and advanced machine learning techniques have been implemented to improve results [13, 25, 32, 43] such as feature selection, preprocessing or model selection. Furthermore, link prediction has been applied to all kinds of domains where networks can be found such as social networks [12, 22, 34] and gene networks [27]. In this paper, we conduct further research into the subject of link prediction.

As previously stated, link prediction is a supervised learning task. To reduce bias in the training of a model, it is common to split the data into a training and a testing subset. If the same data was used to train and test the model, it would likely perform remarkably well, meaning it has memorized the data and therefore cannot generalize to new data. We developed a framework that allows different splits across our datasets to be tested and compared.

With the growth of Machine Learning (ML) and Artificial Intelligence (AI), researchers have developed techniques to improve learning of new tasks. Among those techniques that have gathered high interest, are transfer learning and pre-trained models [18, 33, 44, 47]. In this thesis, we set to investigate transfer learning in a link prediction setting for real-world networks. Works such as Torrey et al. [40] and Tsung et al. [41] refer to transfer learning as an effective framework for tackling new tasks in target domains by transferring previously-acquired knowledge from a related task. Additionally, in recent years, researchers have made important improvements to yield significant performance. Notably, AI has been showing improvements with deep learning, such as the work of Vaswani et al. [42]. In their research, they explore the transformer architecture for convolutional neural networks, which has helped to continuously develop better state-of-the-art techniques in the AI field. Similarly, generative pre-trained AI models have been gaining momentum [11, 36, 37, 38]. These kinds of models are the next step of transfer learning, where a models has already been trained on certain data and has been fine-tuned enough to perform well on new data seamlessly.

In this research, we conduct an investigation of transfer learning for link prediction. This approach seeks to determine how a pre-trained link prediction model for a network can be employed to predict links in other networks. We focus on network topology since it is often given (pre-calculated). Therefore, it can be an early indicator of whether a network will be good for training a predictive model, or a good candidate to have its missing links predicted by a pre-trained model.

Specifically, we set answer to the following questions:

- To what extent can transfer learning be applied to predict unseen links in real-world networks by employing pre-trained models?
    1. How can we efficiently test and compare transfer learning performance across different networks?
    2. What structural network properties yield good transfer learning performance?
    3. What structural network similarities between a train and test set, yield good transfer learning performance?

To answer these questions, we first propose a structure to perform transfer learning of link prediction. Then, we analyse the characteristics and topology of networks to understand how it affects the ability to train or predict links in a network. Moreover, we analyse the pairwise features of both datasets used for training in order to find out what makes for better transfer learning performance.

Through our research, we take the first steps towards automated model selection for link prediction. Based on structural properties of a network dataset, the ideal link prediction model can be selected from a collection of pre-trained models.

The structure of this thesis is as follows. In Section 2 we discuss previous related research on link prediction and transfer learning. Section 3 introduces the prior knowledge, definitions and fundamentals for this thesis. In Section 4, we discuss the approach followed in this thesis and the data used, as well as the algorithms and evaluation criteria. Finally, Section 5 describes the experimental set-up and results of our research, which are then used to draw conclusions in Section 6.

2 Related Work

In this section we present some of the works that influenced our research. These works were chosen either because they define an underlying problem that we are investigating, or because their methods findings presented groundwork for our own research.

With regards to link prediction, Liben-Nowell et al. [23] described one of the first approaches to this problem in social networks, where they model a social network as a graph and identified its edges with a timestamp. They then formulate the link prediction problem, splitting the data at a certain point in time, resulting in two intervals. With this, the algorithm is tasked with outputting a list of edges, not present in the first interval, that appear in the second. Moreover, they detail the methods used to build predictors for their link prediction model. These can be categorised into methods based on node neighbourhoods, based on ensemble of all paths, and higher-level approaches. With this, they showed it is possible to use network topology alone to extract information about future interactions. They also found that simple measures like common neighbours or the Adamic-Adar [1] index perform well, as does the Katz centrality. Nonetheless, they also suggest there was room for improvement in the performance of this task. While our research does not rely on temporal networks to distinguish between train and test sets, this task is very influential and shows the basis of the methodology we aim to follow.

Second, Leskovec et al. [22] built on this research to differentiate positive and negative links. They used features such as Adamic-Adar [1], Jaccard Coefficient, Preferential Attachment and node2vec [17] to train a link prediction algorithm using a two-step path involving two nodes. Additionally, they mention a classifier model used for the task, as well as introducing area under the curve (AUC) and accuracy as evaluation metrics. They determined that *models capture principles that arguably generalize to other domains* (which can be seen as a form of transfer learning, where the model is trained on some data and applied to another). In their work, Leskovec et al. also account for class imbalance for better predictions by resampling the training data, similar to what others [24, 26, 39] proposed. We set to employ methods such as using a two-step path for the feature creation of a pairwise dataset, as well as evaluating the performance using the AUC score, since it accounts for both precision and recall and we believe it will show a competent overview of model performance. We will also take class imbalance into account when training our models.

Third, Ghasemian et al. [16] investigated the use of stacking multiple classification models into a better ensamble classifier, claiming that combining linear algorithms yields better performance and nearly optimal predictions. Like Leskovec et al. [22], they use AUC as the standard evaluator, as well as predictors from topological features (global, node and pair-wise based), model-based predictors and embedding-based predictors. Additionally, they found that link prediction may be fundamentally easier in social networks than in biological or technological networks.

Finally, Bors [9] explored network groupings that effectively generalize the process of selecting good measures as features in link prediction. Contrary to Leskovec et al. and Ghasemian et al., they found that topology-based groupings are better than domain-based groupings to find good measures for link prediction, although they agree with Ghasemian et al. in that social networks yield better performance. While our research points to a generalisation of domains, it is in fact guided by the topological similarities these domains show.

Regarding transfer learning, we follow the work by Torrey et al. [40], which aims to make machine learning as efficient as human learning. They describe methods to transfer knowledge learned in one or more *source* tasks and use it to improve learning in a related *target* task. This way, they attain learning in the target task by leveraging knowledge from the source task. We relate this problem to using different networks as source and target tasks.

We will loosely base the link prediction models on the ones described in the research by Ghasemian et al. [16] and Bors [9]. While our aim is not to produce the most accurate model for a single case, we believe that adapting both of these approaches will result in reliable transfer learning models that show if link prediction can be transferable.

3 Preliminaries

We describe notation and basic network concepts that will be used throughout the thesis in Section 3.1, followed by detailing the network structures we use in Section 3.2. Next, in Sections 3.3 and 3.4 we define the link prediction problem and transfer learning respectively.

3.1 Networks and Graphs

Barabási [4] defines a network as a catalogue of a system's components, the nodes, and the interactions between them, called links or edges. A network can be represented as a graph G, which Bollobas [8] describes as a collection of disjoint sets V and E where $V = V(G)$ is a set of vertices of G and $E = E(G)$ is an *unordered* set of edges. Following [6], formally, a graph is expressed as :

$$G = (V, E)$$

- V, a set of nodes;
- $E \subseteq \{\{u, v\} | u, v \in V \text{ and } u \neq v\}$, a set of edges, where each edge denotes a connection between two different nodes.

An edge linking nodes u and v can be denoted by $\{u, v\}$. If $\{u, v\} \in E(G)$, then u and v are neighbouring nodes of G. Additionally, we call $G' = (V', E')$ a subgraph of $G = (V, E)$ if $V' \subset V$ and $E' \subset E$ and V' includes all endpoints of the E' ($E' \subseteq \{\{u, v\} | u, v \in V' \text{ and } u \neq v\}$).

Moreover, if the edges are *ordered* pairs of nodes, we call the graph directed. On the other hand, a graph with *unordered* edges, where the relation between pairs of nodes does not have a direction, is referred to as undirected. Additionally, a graph can be either static or dynamic, which refers to the formation of the edges in the network. While we have been describing the former, in the latter edges can appear and disappear in a temporal manner. For this thesis, we focus on static, undirected networks.

3.2 Network topology

Several properties exist that characterise the global structure of a network (Barabási [4]). These properties guide us in the exploration of how structure relates to the performance of a link prediction algorithm. Each of the properties is defined on the graph G. Below we describe these properties.

We begin by aggregating two node properties (degree and clustering coefficient) for the whole graph as follows:

- **Degree** Given node u, $\Gamma(u)$ is the set of all neighbouring nodes of u. The size of this set $|\Gamma(u)|$ is called the degree of node u and is represented as $k(u)$. It is also known as

the number of connections involving node u.

- **Average degree** Given the degree of all nodes, we calculate the average degree as:

$$\bar{k} = \frac{1}{|V|} \sum_{u \in V} k(u)$$

- **Maximum degree** The size of the set of neighbours for the node with the most neighbours.

$$k_{max} = max_{u \in V} k(u)$$

- **Clustering coefficient** The clustering coefficient captures the degree to which neighbours of a given node link to each others. For a node u with degree $k(u)$, and $L(u)$ links between its neighbours, the local clustering coefficient is defined as:

$$c(u) = \frac{2L(u)}{k(u) \cdot (k(u) - 1)}$$

We aggregate the clustering coefficient as the average of all nodes, which is often referred to as the *local clustering coefficient* c.

$$c = \frac{1}{|V|} \sum_{u \in V} c(u)$$

Then, we calculate the following global properties.

- **Number of nodes** The number of objects in the system. $n = |V|$
- **Number of edges** The total number of connected nodes. $m = |E(G)|$
- **Diameter** Given the shortest path between two nodes, the diameter of a network is the maximum shortest path length in the network.
- **Degree assortativity** The degree assortativity ρ in a network is defined as the Pearson correlation coefficient of the degree of connected nodes. A network is said to have assortativity when high-degree nodes are connected to other high-degree nodes; whereas in a network with disassortativity, high-degree nodes and low-degree nodes are typically connected to each other. The assortativity is calculated as:

$$\rho = \frac{\sum_{\{u,v\} \in V} (k(u) - \bar{k}) \cdot (k(v) - \bar{k})}{\sum_{\{u,v\} \in V} (k(u) - \bar{k})^2}$$

- **Gini coefficient** The Gini coefficient g is a topology feature proposed by Kunegis et al. [19]. It is based on the Lorenz Curve as seen in Figure 1 and it is a measure from economics used to measure inequality. In this case it is applied to degree distribution. It can be calculated as:

$$g = \frac{\sum_{\{u,v\} \in V} |k(u) - k(v)|}{2 \cdot |V|^2 \cdot \bar{k}}$$

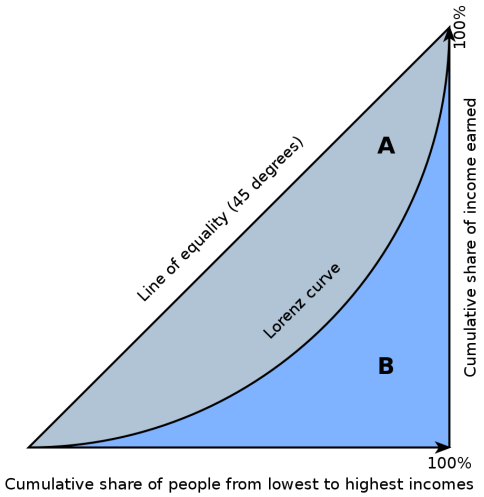


Figure 1: Graphical representation of the Gini coefficient in relation to the Lorenz Curve using an example from economics. The Gini coefficient is equal to the area marked A divided by the sum of the areas marked A and B, that is, $G = \frac{A}{A+B}$ (Image [46]).

- **Transitivity** The transitivity C is defined as the probability that two incident edges are completed by a third to form a triangle, It is often also called *global clustering coefficient*. Given, the number of triangles in the graph t and the number of connected triples of vertices s, it is calculated as:

$$C = \frac{3t}{s}$$

3.3 Link prediction

Link prediction is a method that seeks to solve the problem of missing links in a network, either because the given snapshot is incomplete, or because the network is dynamic (changes through time). It can be referred to as a supervised learning task. That is, a type of machine learning task that involves training a model on a labeled dataset, where the labels are the correct outputs for each input. In supervised learning, the goal is to learn a mapping between input data (node-based topological features of a pairs of nodes) and the corresponding output data (whether a link between two nodes exists or not), given a set of training examples. Further, as Liben-Nowell et al. explained —the link prediction problem asks *to what extent*

can the evolution of a social network be modeled using features intrinsic to the network itself. Hence, link prediction seeks to accurately predict the edges that will be added to the network. In practice, this can be due to different reasons such as an incomplete graph, or a static snapshot of a dynamic temporal network.

We define link prediction as the task of learning to identify potential missing links in an incomplete, static network.

3.4 Transfer learning

For the purposes of this research, we refer to transfer learning as a method in which a machine learning model is trained to learn how to predict the missing links for a specific network and then using that same model to predict undetected links in a different, unseen network.

4 Methodology

In this section, we describe in depth the steps taken to answer the questions posed in Section 1. Figure 2 shows an overview of the process. We will detail all the steps throughout. In Section 4.1 we first explain the data used in our research. Next, we describe how the link prediction model is built, including how we set up a node pair dataset, in Section 4.2. Later, in Section 4.3, we present our approach to training models across datasets. Finally, we detail our approach to measuring model performance in Section 4.4.

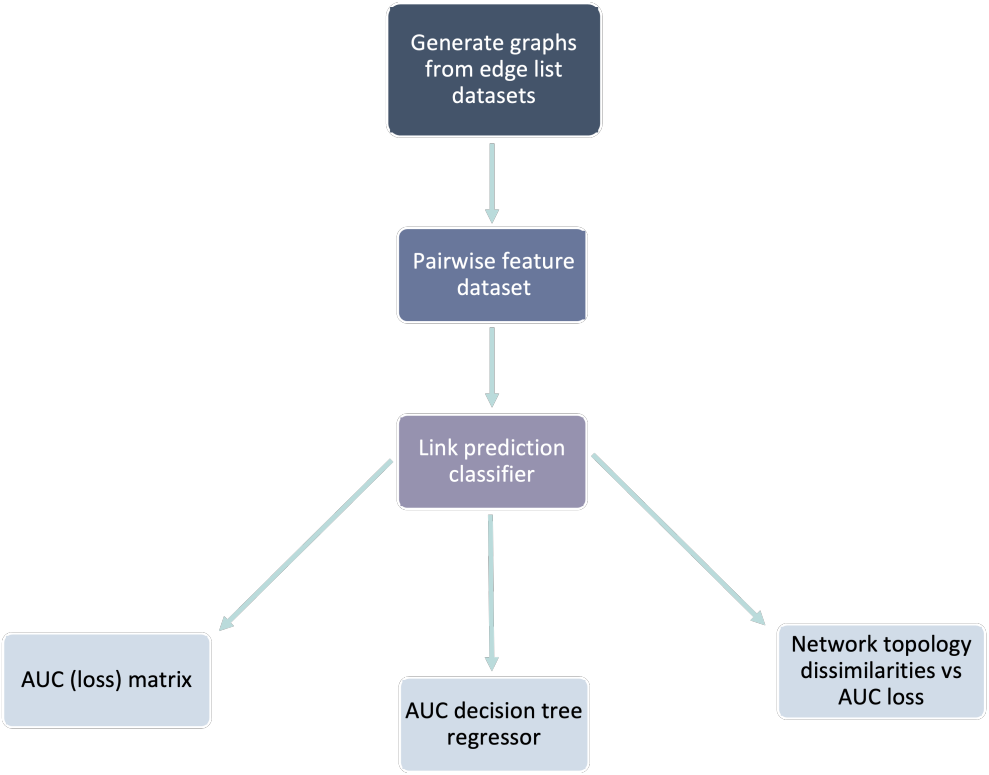


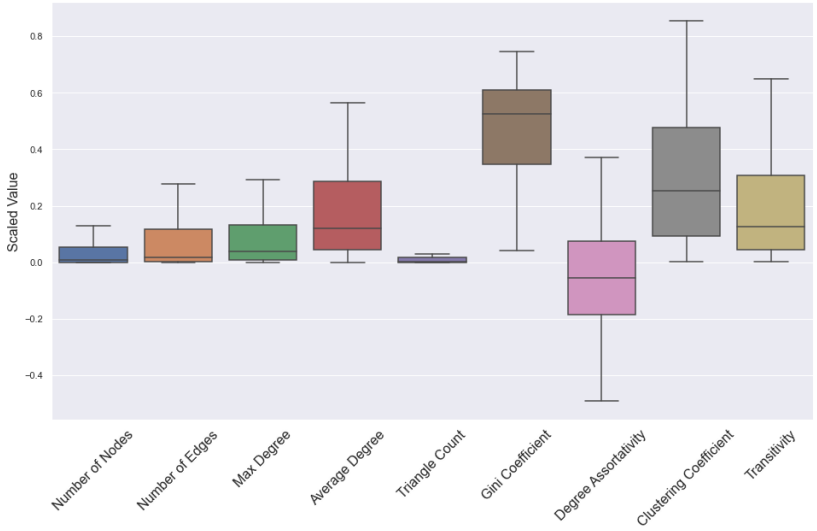
Figure 2: Overview of the experimental pipeline. The *AUC (loss) matrix* is used to answer the first research question. The second question is addressed by the *AUC decision tree regressor*. And the *network topology dissimilarities vs AUC loss* analysis will help answer the third question.

4.1 Data

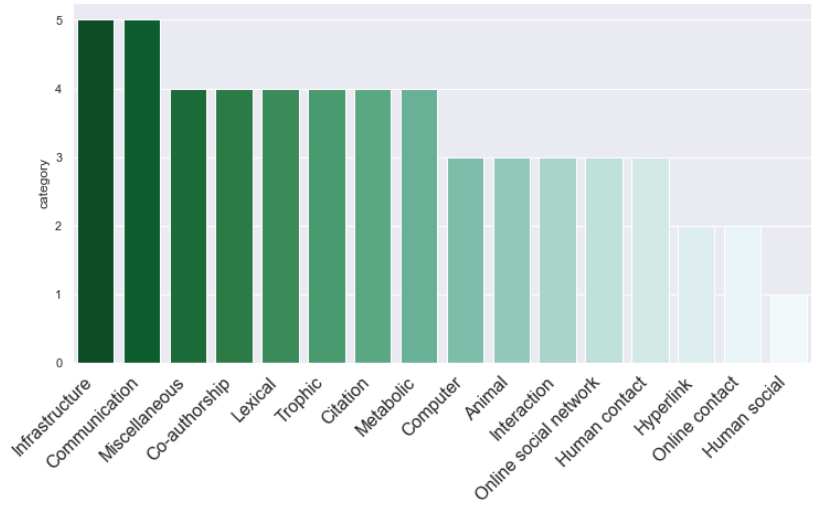
In order to test the capabilities of transfer learning, we need a wide variety of datasets. With such a wide variety of datasets, we can thoroughly test how different structural networks and properties affect the ability to perform transfer learning. Data was gathered from the KONECT Project [20]. Once the data was retrieved, we performed pre-processing. To be able to compare the networks, we interpret them as undirected, unweighted and unipartite graphs. Additionally, we defined the network to contain only the giant connected component, a subgraph of the original network in which no nodes are isolated. This subgraph contains the vast majority of nodes and ensures computation of statistical features, that require the

graph to be connected.

In total, we gathered 49 datasets. Some basic statistics are available in Table 6. The networks were chosen such that they cover a variety of topological properties, sizes and categories (as reported in KONECT Project [20] and shown in Figure 3 . We excluded the biggest networks since it would make the experiments too computationally expensive. Note that, as shown in Figure 3, we gathered a diverse sample of datasets in favour of better test results and understanding how and when transfer learning works well.



(a) *Properties distribution.* Properties are scaled (outliers are not shown because they created a lot of noise).



(b) *Category distribution.* (cf. KONECT project [20]).

Figure 3: Network properties and categories distributions.

4.2 Link prediction model

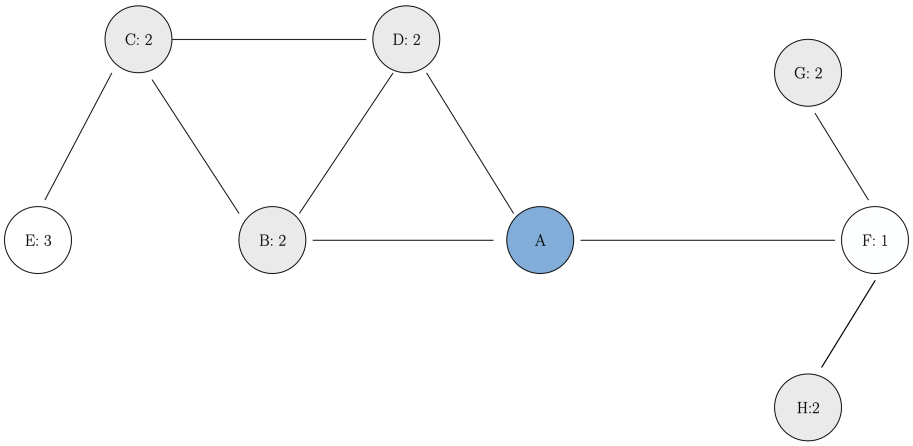
In this section, we first detail how we create a tabular dataset to train the predictive model based on a network, since a network cannot be directly input into these algorithms. This results in a dataset for each network. Next, we describe the features we compute for the dataset, which will be used as predictors for the model. Then, we give an overview of the machine learning algorithm used to predict missing links and we explain how we deal with splitting the datasets for training and testing.

4.2.1 Pairwise training dataset

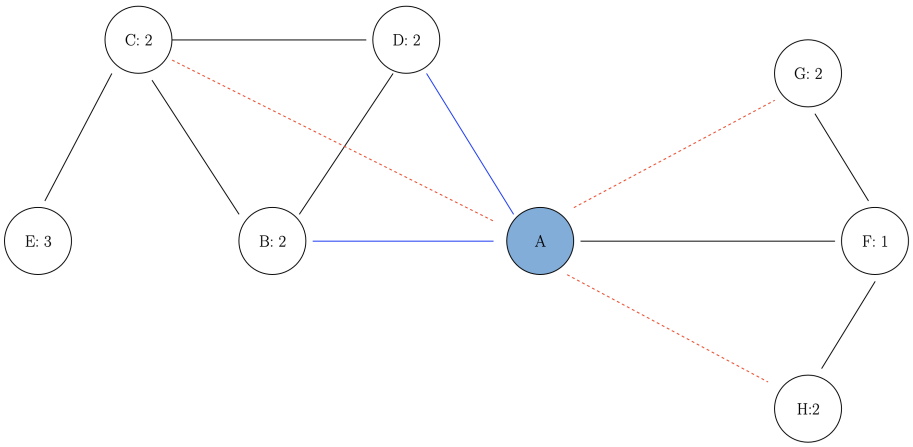
It is a common approach to construct a training dataset and use network statistics of pairs of nodes to distinguish between positive and negative cases of connected nodes. For this, we sample existing pairs of nodes from the network, creating a dataset to input to the training and prediction models. In every network, we report each node as a source and all nodes that can be reached with a walk of distance two as a target, as shown in Table 1 and Figure 4. Next, we proceed to add a label to node pairs that are already connected at a distance of one as to distinguish between positive and negative cases. Lastly, we compute features for each pair of nodes, which are detailed in Section 4.2.2. We do this for each network.

Source	**Target**	**Label**
A	B	1
A	C	0
A	D	1
A	G	0
A	H	0

Table 1: Link prediction training dataset of nodes at distance two of A.



(a) Original network. Nodes at a walk of distance two from A (B, C, D, G and H).



(b) Link predictions candidates. Blue links would be positive cases, whereas red links are negative ones.

Figure 4: Distance two walk from node A. For node A, all nodes but E and F can be reached by a walk of distance two. Then, node A is already connected to nodes B and D, thus labelling those as existing links. This is captured in a dataset, represented in Table 1.

4.2.2 Features

The basis of training a machine learning model for link prediction, is to understand why a pair of nodes might be linked or not. Features are calculated for the model to discriminate between positive and negative cases of links between node pairs.

We employ features commonly used in link prediction models, with a focus on the work presented by Bors [9], to come up with a good enough model to test transfer learning. Therefore, once we compute pairs of nodes to generate datasets for each network, we then inspect the similarity of the nodes in each pair. The features are selected in a heuristic approach to balance simplicity, speed and performance.

Common neighbours When nodes are linked to the same neighbours, they are likely to be

similar. For this, the simplest measure is the number of common neighbours shared by two nodes. It is defined by the size of the set of the intersection of the neighbours of two nodes:

$$CN(u, v) = |N(u) \cap N(v)|$$

Jaccard Coefficient Similar to common neighbours, the Jaccard coefficient normalises for the number of unique neighbours of both nodes. This is to account for the fact that nodes with higher degree are more likely to have more common neighbours. It is defined as:

$$J(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

Adamic-Adar [1] Considering common neighbours again, Adamic-Adar prioritises nodes with low degree. It is based on the idea that when predicting links, linking a node to a large hub is not as meaningful as linking to an isolated node. It is calculated as:

$$AA(u, v) = \sum_{x \in N(u) \cap N(v)} \frac{1}{\log(k(x))}$$

Preferential Attachment [23, 31] The basic premise, is that the probability that a new edge involves node u is proportional to $|N(u)|$, the current number of neighbors of u.

$$PA(u, v) = |N(u)| \cdot |N(v)|$$

Node degree The total number of neighbours of a node. We record both the degree of the source and the target nodes, as well as a ratio to account for difference in the hopes of identifying isolated nodes and hubs. For a node u with a set of neighbours $N(u)$:

$$k(u) = |N(u)|$$

$$k(u, v) = \frac{k(u)}{k(v)} = \frac{|N(u)|}{|N(v)|}$$

Triangle count The number of triangles that a node participates in.

$$t(u) = \frac{|\{\{u, v, w | u \leftrightarrow v \leftrightarrow w \leftrightarrow u\}|}{6}$$

4.2.3 Stacked classifier

Once we have created a pairwise training dataset and calculated its features, our supervised learning link prediction classifier is trained (third step of Figure 2). Based on the findings by Ghasemian [16], we define a stacked classifier using *Scikit-Learn* [35] as follows:

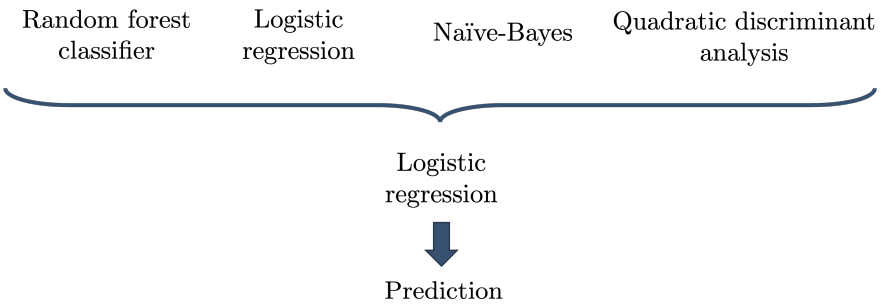


Figure 5: Stacked classifier.

These classifiers were selected because they are some of the most commonly used classifier models in the field. After testing them separately, we observed sufficient performance.

4.3 Cross-validation training

As it is common practice, we split the datasets into training and testing subsets to be able to evaluate the performance of the prediction model. The datasets are split using the *Scikit-Learn k-fold* function [35]. In this case, we set $k = 4$, resulting in a random 75–25% split, to avoid bias in the sample.

Additionally, due to the sparsity of networks, it is common for the node pair dataset to have class imbalance which could limit the algorithm's learning ability. To account for this, we down-sample the majority class of the training dataset as shown in Figure 8. However, down-sampling is not necessary in the test set since we seek for the algorithm to learn to distinguish between positive and negative cases as much as possible, and removing information in this step could hinder the performance.

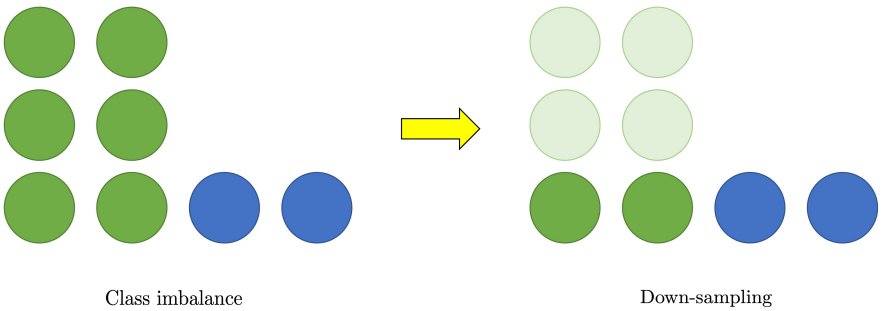


Figure 6: Down-sampling for class imbalance.

Since the *k-fold* split is shuffled randomly, we cannot guarantee that the results (either good

or bad) are due to randomness. Hence, we perform cross-validation training as shown in Figure 7 to validate a model's results more than once. Essentially, we train a model for each fold per dataset. Next, we validate the models on each split of the datasets. Note that in Figure 7, when testing on the same network, we only train and validate on the same split. This is because within the same split, the data is disjoint, so we do not have the same observations in the train and validation sets. Otherwise, we would encounter the same data in both the training and validation sets, which is not ideal for a machine learning model, since it adds bias to the model by predicting previously seen data. Thus, we remove those cases in our testing set.

Lastly, we evaluate the performance of each model with each validation set and append AUC scores (explained in Section 4.4) to a matrix. With this set-up, we lay the groundwork to answer our first research question.

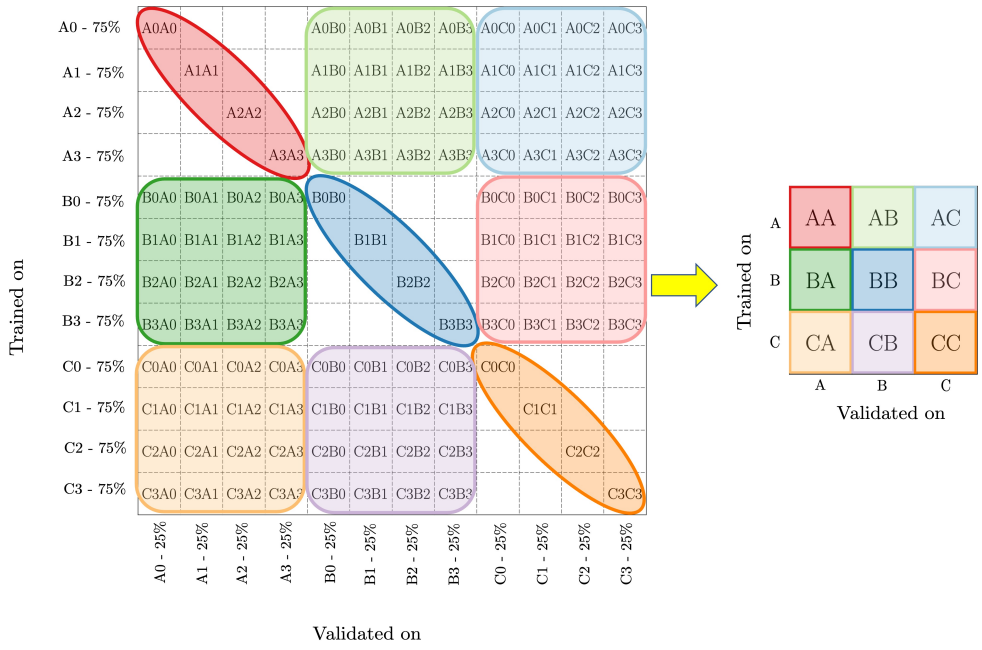


Figure 7: Cross-validation training assignments. For each split, we generate a training set with 75% of the data and a validation set with the remaining 25%. (Letters indicate the dataset, numbers indicate the split.)

4.4 Performance metrics

Similarly to previous work [9, 13, 16], we measure the performance of the classifier by means of the Area Under the Receiver Operating Characteristic Curve (AUC), which will be discussed below. Additionally, we define AUC loss, which we use to measure the loss in accuracy resulting from doing transfer learning between two different networks.

Area under the ROC curve (AUC)

To evaluate the capabilities of transfer learning for link prediction, we need a metric to measure the performance of a model on the validation set. For this, we use the area under the receiver operating characteristic (ROC) curve (AUC). Proposed by Bradley et al. [10], it is a good single performance metric for classification tasks where classification models output counts of the correct and incorrect classifications from each class. This metric is based on the confusion matrix (Table 2), which is used to show the performance of a classification algorithm.

		Predicted condition	
	Total population = P + N	**Positive (PP)**	**Negative (PN)**
Actual condition	**Positive (P)**	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Table 2: Confusion matrix [45].

Then, using the results from the confusion matrix, we can calculate the recall or true positive rate (TPR) and false positive rate (FPR) as:

$$TPR = \frac{TP}{TP + FP} \qquad FPR = \frac{FP}{FP + TN}$$

The resulting rates are both in the range $[0, 1]$, where $TPR \approx 1$ and $FPR \approx 0$ imply better model performance. By plotting both together as seen in Figure 8, we can calculate the receiver operating characteristic (ROC) curve. Then, the area under the ROC curve (AUC) is calculated and used as an overall measure of the performance of a predictive model. Higher AUC scores indicate the model is better at predicting the correct labels [29]. Due to its versatility and interpretable results, the AUC is a common measure when scoring classifier models [9, 13, 16].

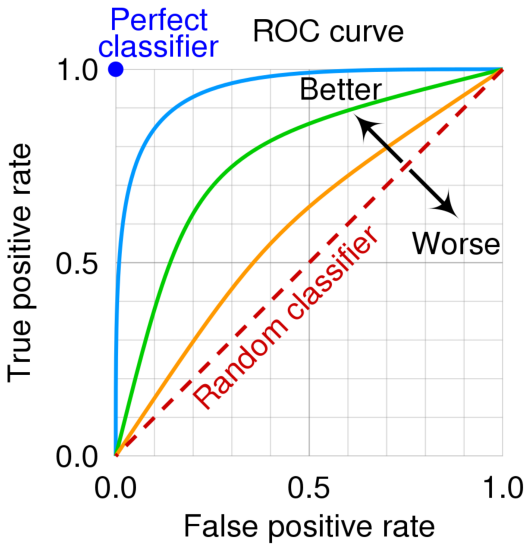


Figure 8: Three ROC curves for different models [9].

Once we have trained predictive models for each network, we set to use those same models to validate link prediction across our set of datasets as shown in Figure 7. This will generate an AUC score for every ordered pair of datasets, where one dataset is used for training and the other one for validation, allowing us to assess to what degree transfer learning is possible for link prediction. Once we have all pairs of scores, we can proceed to analyse and visualise these results.

AUC loss

Additionally, while the AUC score is useful to get an overview of the performance of the models, we also need to inspect how transfer learning affects the score. For this we calculate the loss in AUC from the validation set compared to the train set. The idea behind this is that the smaller the AUC loss, the better the model is at capturing the knowledge required to correctly predict links in a different network. This will allows us to make a pairwise comparison of train and validation datasets that we can then use to analyse along with both network's topological properties. We calculate loss $\mathcal{L}_{i,j}$ as follows:

$$\mathcal{L}_{i,j} = AUC_{i,i} - AUC_{i,j}$$

Where, $\mathcal{L}_{i,j}$ is the loss of training a model with network i and validating on network j; and $AUC_{i,j}$ is the performance score of training a model with network i and validating on network j.

5 Experiments and results

In this section we outline the setup of our experiments and review results. First, Section 5.1 describes the experimental set-up and environment used. Then we discuss our findings in Section 5.2, starting with an analysis of performance in Sections 5.2.1 and 5.2.2, where we discuss the extent to which transfer learning was achieved in our experiments. Next, we compare transfer learning and its relation to topological features in Section 5.2.3, and to topological dissimilarities in Section 5.2.4. Lastly, Section 5.2.5 presents a case study of two pairs of networks, where we examine the features of the datasets created from the pairs of nodes in a network. The goal of this case study is to detect whether the features of the pairwise dataset features play a role in the effectiveness of transfer learning.

5.1 Experimental setup

In this section we detail the procedure to obtain and prepare the data that is used as input for the link prediction models. As mentioned before, we use the Python package Scikit-Learn [35] to train our stacking classifier, as well as other pre-processing techniques needed to prepare the data for the development of the predictive models.

Following the approach of Ghasemian et al. and Bors, we fit individual models, although in this case for every split of each dataset, resulting in 196 models with the same feature set. The parameters for each estimator of the stacking classifier are as follows:

- RandomForestClassifier: `n_estimators = 10`
- LogisticRegression: *default*
- GaussianNB: *default*
- QuadraticDiscriminantAnalysis: *default*

5.2 Results

Below, we analyse the experiments and results used to answer our research questions. We first discuss the results from the AUC scores and loss matrices in Sections 5.2.1 and 5.2.2 respectively. Next, we investigate the feature importances in Section 5.2.3 and later we examine the impact of dissimilarities between training and validation networks. Lastly, in Section 5.2.5 we study the effect of similarity between the pairwise features of different networks on transfer learning.

5.2.1 AUC scores matrix

In this section, we set to examine how to test and compare transfer learning, i.e., we answer the first research question proposed in Section 1, which we continue in the following section. For this, we train and test link prediction classifiers across all the networks to examine their performance.

Figure 9 shows the heatmap resulting from the cross-validation training procedure detailed in Section 4.3. It displays the AUC scores from training a model on a network and using that same model to predict missing links in another network. Rows in the matrix depict training performance, while columns represent ease of prediction. There, we can see that most of the AUC scores are above 0.5, which means the models do perform better than a random classifier which assigns 0 or 1 with the same probability. Moreover, the average AUC score is **0.71**, which suggests a good performance overall.

Additionally, we can already see that networks from the citation and co-authorship categories (like `cora`, `dblp`, `hpph`, `hepth`, `astroph` and `astrophysics`), as well as miscellaneous (`asoif`, `sistercities`, `lesmis`), show favourable training performance, suggesting they are good baseline for pre-trained models. Similarly, computer networks (`caida`, `gnutella25`, and `routeviews`), infrastructure networks (`airtraffic`, `newyork`, `euroroad`) and metabolic networks (`proteins`, `yeast`) display good validation performance, meaning they are usually easy to predict. On the other hand, the bible network is the worst performing training network. Also, some human contact (`residence`, `karate`), human social (`adolescent`), lexical networks (`bible`, `eat`, `wordnet`) and trophic[1] networks (`florida_dry`, `florida_wet`, `littlerocklake`, `chesapeake`) show substandard validation performance.

[1] From The KONECT Project. Relating to biological interactions of species, commonly food chains.

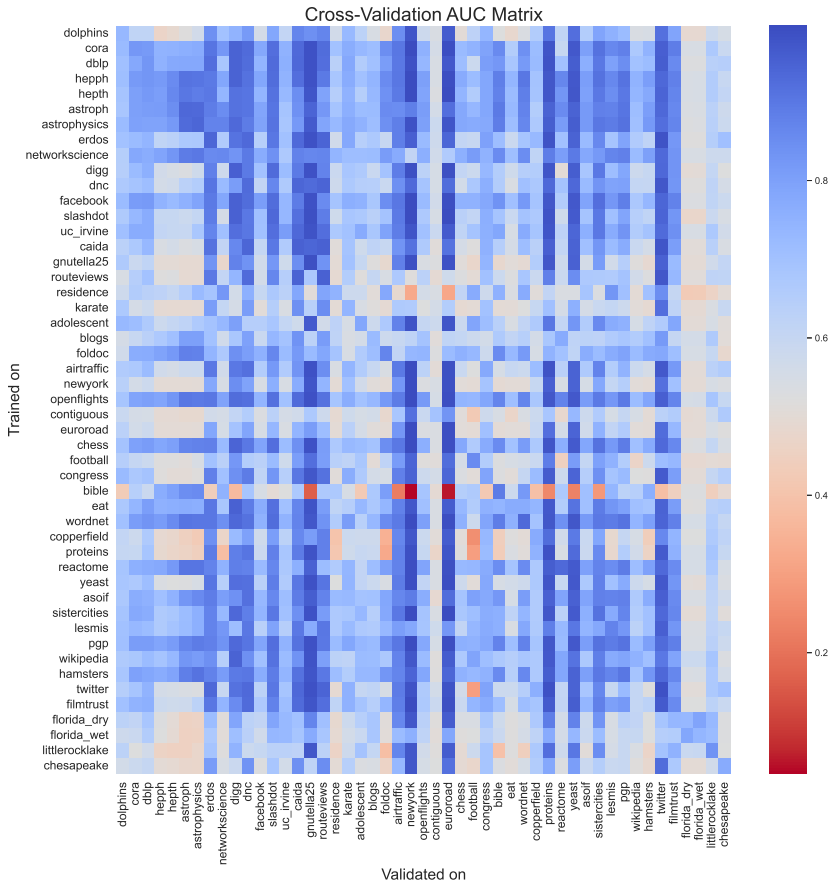


Figure 9: AUC scores matrix. Networks are ordered according to their category.

Furthermore, Figure 10 shows the distribution of AUC scores across al pairs of networks. The results are in line with what was mentioned before, since we can clearly see that most scores are greater than 0.5 and a median of **0.71**.

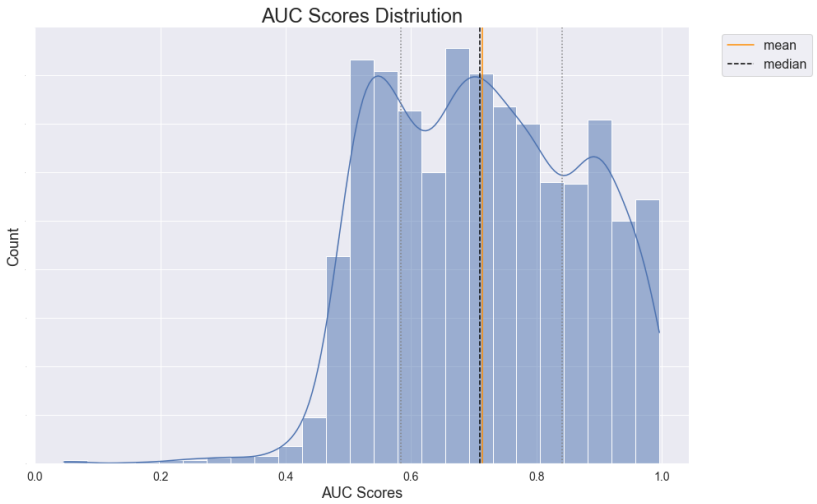


Figure 10: Distribution of AUC scores across all networks. Mean and median are highlighted, while the trend is shown in blue and the quartiles in light grey.

Table 3 presents the aggregated AUC scores by category, showing the average and maximum scores for each category, both for training and validating. This means we can examine which categories might be better to train a missing link prediction model, and which are better at being predicted. We can see that citation and co-authorship networks show the best training performance, and computer and metabolic networks the best validation performance, which is in line with previous results. Moreover, human social and trophic networks are also the worst performing both in terms of training and validation, since they show the lowest scores. This is likely because of their low degree assortativity and Gini coefficient.

Category	**Training Average AUC Score**	**Validation Average AUC Score**	**Training Max AUC Score**	**Validation Min AUC Score**
Animal	0.6698	*0.6583*	*0.86*	*0.732*
Citation	0.7967	0.6698	0.9245	0.8042
Coauthorship	0.7791	0.7172	0.8904	0.8541
Communication	0.7392	0.7666	0.927	0.8564
Computer	0.6619	0.854	0.8972	0.9415
Human Contact	0.6085	0.6398	0.7351	0.7192
Human Social	0.6823	0.6523	0.8022	0.7839
Hyperlink	0.68	0.649	0.7669	0.7786
Infrastructure	0.6676	0.7611	0.8088	0.8382
Interaction	0.7072	0.6586	0.883	0.745
Lexical	0.686	0.6569	0.7823	0.7811
Metabolic	0.7077	0.8136	0.8849	0.8963
Miscellaneous	0.7558	0.7485	0.8808	0.8468
Online Contact	0.7777	0.6949	0.9018	0.7982
Online Social Network	0.7451	0.7734	0.9415	0.8605
Trophic	0.6219	0.5629	0.7458	0.6516
Average	*0.7054*	*0.7048*	*0.8520*	*0.8055*

Table 3: Performance results by category[2]. In each column, the top two results are highlighted in green, while the bottom two results are highlighted in red.

5.2.2 AUC loss matrix

Further, in this section we continue to investigate the efficiency of transfer learning for link prediction. In this case, rather than comparing the raw performance of the models, we set to compare how effectively they manage to retain information and predict missing links in a different network.

Similar to the last section, Figure 11 is constructed based on AUC loss scores. Since we are comparing two *different* networks, we ignore and remove the diagonal from the matrix. In the end, we have $(N \times N) - N$ loss scores where the lower the score, the better, since it means that we had a lower loss in performance, going from validating on the same dataset for which the model was trained to validating in a different dataset. The overall individual conclusions we can draw from this are similar to the ones in the previous sections, where the same networks and categories seem to perform alike. Nonetheless, this analysis is relevant for further investigations into the performance when dealing with pairs of networks as in Section 5.2.4.

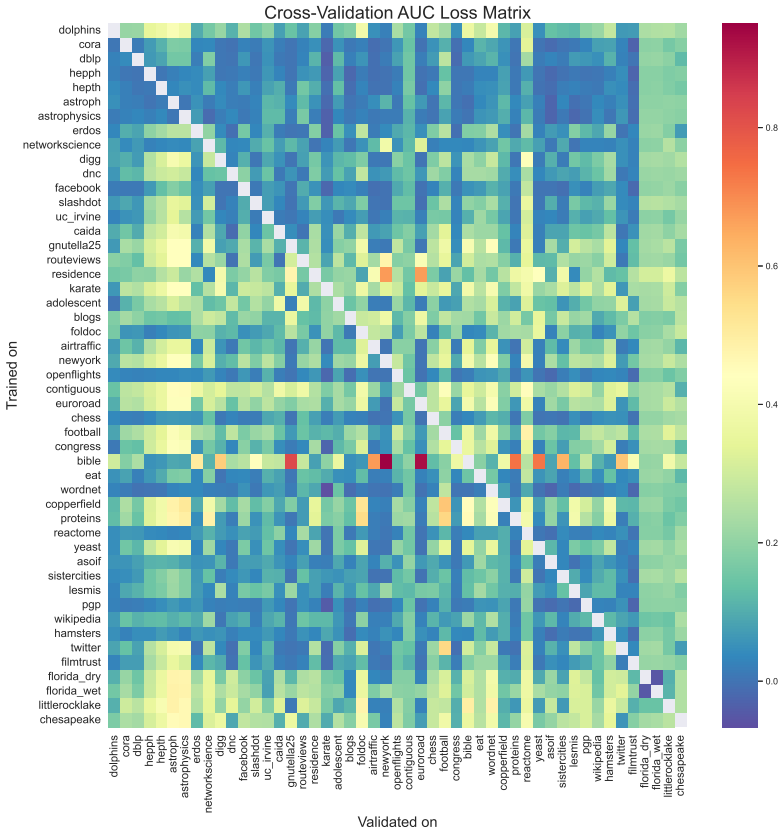


Figure 11: AUC loss matrix.

[2] Additional information like the number of datasets per category can be found in Table 5.

As for the distributions of the AUC loss shown in Figure 12, we can still see that with an average of 0.14 and median of 0.12 the loss is sufficiently low to consider adequate.

Figure 12: Distribution of AUC loss across all networks. Mean and median are highlighted. Trend is shown in blue, quartiles in light grey.

Additionally, we present Table 4, where we aggregate by category similar to in the last section, only this time for AUC loss, where we have to take into account that the lower the loss, the better. Again, the results are in line with the ones discussed in the section above. However, in this case, co-authorship networks still perform among the best when it comes to training, but among the worst in terms of validation. This is likely because when the training and validating with networks of this category, the scores are quite high, whereas training with other categories and validating with co-authorship networks, the scores are still relevant, but not as high as in the former case.

Category	Training Average AUC Loss	Validation Average AUC Loss	Training Min AUC Loss	Validation Min AUC Loss
Animal	0.1909	0.0789	0.0991	0.0253
Citation	0.0518	0.1626	0.0121	0.0368
Coauthorship	0.0702	0.203	0.0317	0.085
Communication	0.1095	0.1197	0.0346	0.0291
Computer	0.1881	0.1097	0.0965	0.0201
Human Contact	0.241	0.1116	0.1103	0.0257
Human Social	0.1715	0.1204	0.0045	0.0558
Infrastructure	0.1818	0.1189	0.0899	0.0403
Interaction	0.1451	0.1537	0.0442	0.0629
Lexical	0.165	0.1863	0.1201	0.059
Metabolic	0.1432	0.1513	0.0711	0.0657
Miscellaneous	0.0934	0.1117	0.0431	0.0121
Online Contact	0.0728	0.1379	0.0367	0.032
Online Social Network	0.105	0.1185	0.0201	0.0291
Trophic	0.2286	0.2088	0.1332	0.1566
Average	*0.1425*	*0.1420*	*0.0681*	*0.0476*

Table 4: Loss results by category[3]. In each column, the top two results are highlighted in green, while the bottom two results are highlighted in red.

5.2.3 Tree-based algorithms for topology importances

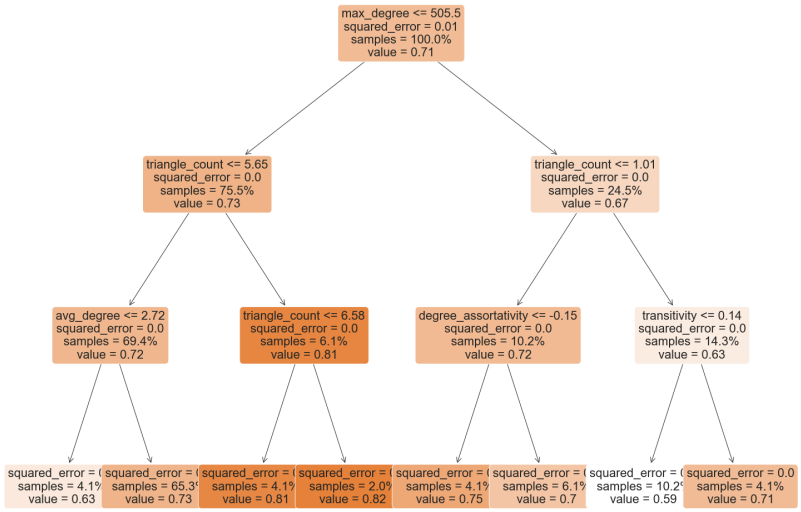
Next, we set to understand how the topological features might affect the *training* performance of a network, i.e., we answer the second research question. To do so, we train a basic decision tree and random forest algorithms by using a network's topology to fit the average AUC score from Section 4.4. Then, we proceed to investigate the decision tree and random forest by visualizing the top splits (discriminating decisions). This helps us understand which are the most important topological features for transfer learning and how the values for these features affect the performance of transfer learning.

Figure 13 shows the resulting tree fitting the topological features of networks to the average AUC score as discussed in Section 4. We explored the tree at a low depth in order to be able to inspect the most important features for transfer learning.

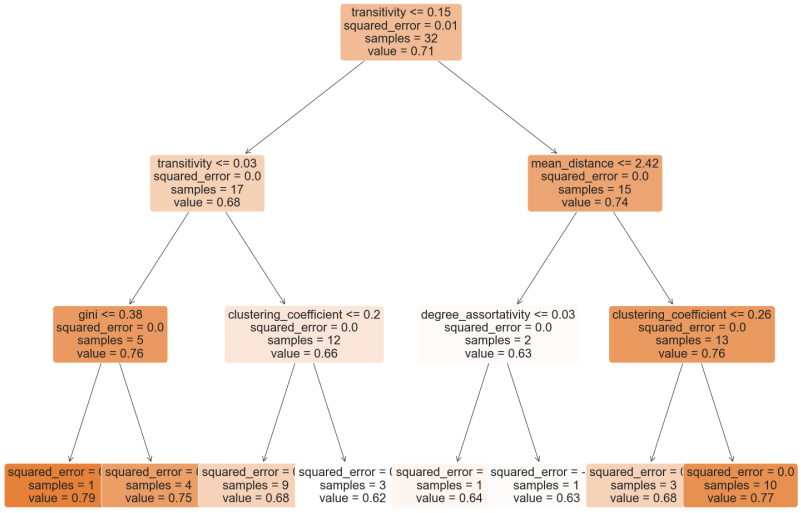
When it comes to the decision tree, we can see that a high number of triangles (per link) appears to be very indicative of high performance, whereas low triangle count and low transitivity seem to be causing low performance.

On the other hand, the random forest regressor suggests that low transitivity, low clustering coefficients, and negative degree assortativity tend to result in lower performance. On the other hand, a high clustering coefficient seems to be important for high performance, as long as mean distance is low. (For high performance, clustering coefficient and transitivity are usually correlated, so as long as a network has high clustering coefficient, it likely will not have low transitivity).

[3] Additional information like the number of datasets per category can be found in Table 5.



(a) Decision tree



(b) Random forest tree

Figure 13: Tree-based topology importances.

Using the most important topological features, we can now analyse the average AUC training scores. Table 5 shows the relation between average AUC scores for each category, and the average of their most relevant topological features. We can see that categories with both high clustering and Gini coefficients tend to perform better. Additionally, triangle count, clustering coefficient and transitivity show some correlation with the training performance. Nonetheless, there appears to be nothing that implies that a category may perform better or worse based on their average topologies.

Category	Training Average AUC	Triangle Count	Clustering Coefficient	Transitivity	Average Degree	Gini Coefficient	Degree Assortativity	Category Size
Animal	0.6698	2.2785	0.5843	0.5998	9.6234	0.2583	0.142	1
Citation	0.7967	2.251	0.2459	0.111	16.448	0.5664	-0.0345	4
Co-authorship	0.7791	3.6671	0.5419	0.3023	11.5925	0.5488	0.0579	4
Communication	0.7392	0.8163	0.0932	0.0485	7.5924	0.6522	-0.0626	5
Computer	0.6619	0.4066	0.1553	0.0071	4.2477	0.5922	-0.1833	3
Human contact	0.6085	0.8582	0.325	0.1981	7.8625	0.3229	-0.2197	2
Human social	0.6823	0.449	0.1467	0.1419	8.2355	0.2899	0.2513	1
Hyperlink	0.68	3.5924	0.3291	0.1696	20.5262	0.4645	-0.1167	2
Infrastructure	0.6676	1.1937	0.2184	0.1558	4.9783	0.3475	0.0958	5
Interaction	0.7072	1.225	0.2793	0.2174	10.3659	0.3908	0.0645	3
Lexical	0.686	1.4976	0.3938	0.1139	13.2331	0.5486	-0.0729	4
Metabolic	0.7077	7.2165	0.1733	0.1665	14.4818	0.5366	-0.159	3
Miscellaneous	0.7558	1.0493	0.2771	0.2057	4.8571	0.5184	-0.0957	3
Online contact	0.7777	4.1429	0.2038	0.2517	16.5235	0.669	0.0775	2
Online social	0.7451	1.3985	0.2627	0.1465	7.539	0.5627	-0.1501	3
Trophic	0.6219	3.4966	0.3609	0.3107	25.1591	0.3102	-0.2142	4

Table 5: Average scores and statistics by category. For every column, higher values are shown in green, while lower values are shown in red.

5.2.4 Dissimilarities vs. AUC loss

In this section, we examine what structural network similarities between a train a test set, lead to good transfer learning performance, i.e., the third research question. We do so by comparing the AUC loss to how dissimilar the topological properties are between two networks.

In order to analyse the ability to do transfer learning between a model and a different dataset, we look at the topologies of the training and validation networks. We do so to understand if there is a relationship between their similarity and the performance of the model. For this, we compare how dissimilar two networks are by their topological properties, to the loss in performance when performing transfer learning between both networks. This allows us to understand what properties to look for in a pair of networks when trying transfer learning. The features we investigated are the following:

- Number of nodes
- Number of edges
- Number of triangles (per link as to normalise for larger networks)
- Maximum degree
- Average degree
- Diameter
- Gini coefficient
- Degree assortativity
- Clustering coefficient
- Transitivity

Smaller dissimilarity values indicate more resemblance between a pair of networks. Since the dissimilarity is calculated (by nontrivial normalization) on a per-feature-basis, each pair of networks has a set of dissimilarity values independent form each other.

Figure 14 illustrates the relation of loss in performance when predicting missing links in one network using a model trained for another, compared to the topological dissimilarities of both networks. It is clear that there is a trend in almost all topological features (except for maximum degree), where the more different two networks are, the more loss in performance is likely.

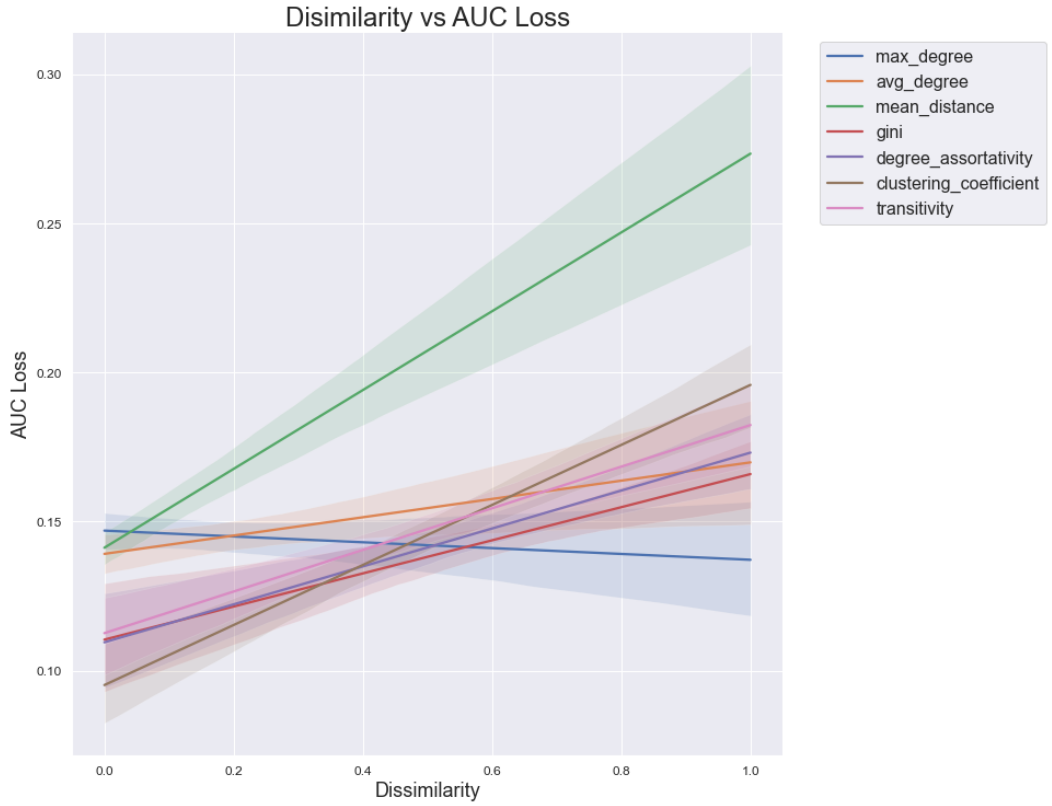


Figure 14: AUC loss vs feature dissimilarity.

5.2.5 Pairwise feature distributions case study

Finally, we study other possible biases in the experiments. Namely, we set to explore pairs of networks based on their node-based datasets used to train the predictive models (which were explained in Section 4.2), as opposed to their topological properties. While there are many different cases (one for every pair of networks), in this section we study only two: a positive and a negative one, where we observed low and high AUC loss respectively.

First, we picked a pair of networks with low loss, `astrophysics` and `networkscience`; and another pair with high loss, `bible` and `newyork`. Then, we train a simple random forest classifier, similar to the one used in the stacking classifiers, for which we were able to calculate

the feature importance, as to get an overview of which features were worth inspecting. These are shown in Figure 15. As we can see, Adamic-Adar was the most important feature for this model.

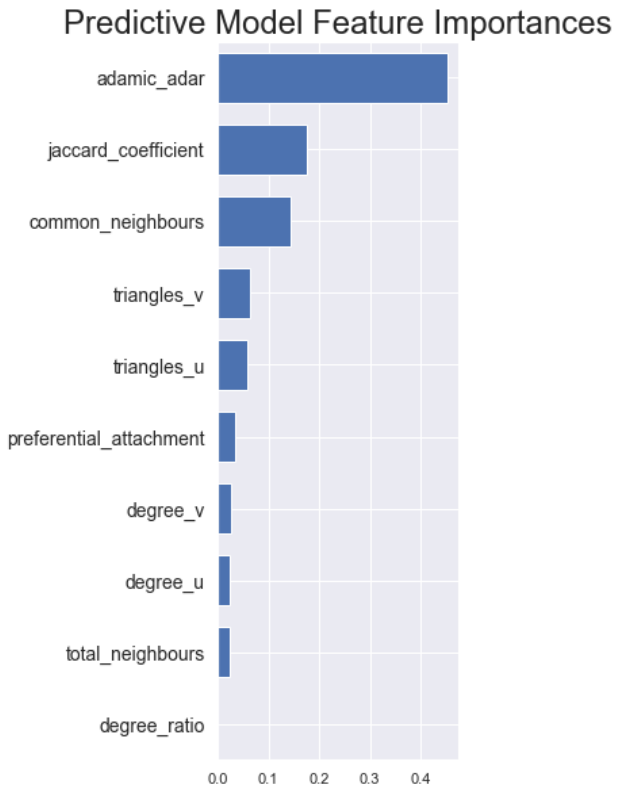
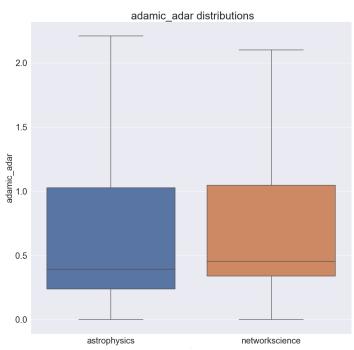
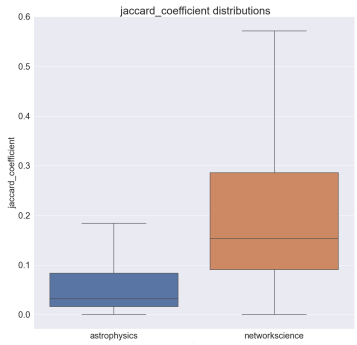


Figure 15: Feature importance of a model trained to predict the missing links in the `astrophysics` network.

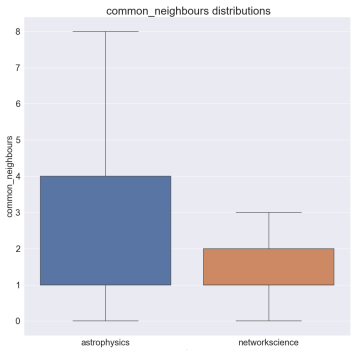
Next, we compared the distributions of both pairs of datasets, considering that two datasets with similar distributions might result in lower loss in performance. For this, we considered statistical tests such as the Mann-Whitney U and Kolmogorov-Smirnov tests. However, the samples used were considerably larger than what is suggested for these tests. Thus, we decided to present a visual comparison of the distributions in the form of box-plots, as seen in Figure 16. This method is helpful to visualize the distribution of one feature of two datasets side by side and analyse relevant statistics that the box-plot provides such as mean, quantiles and the length of the legs (to consider outliers).



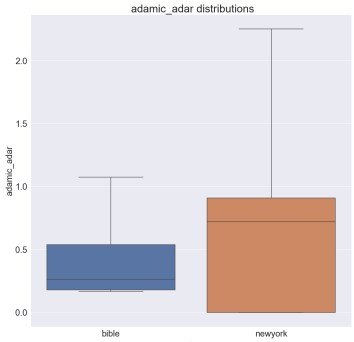
(a) Adamic-Adar of **good** performing networks.



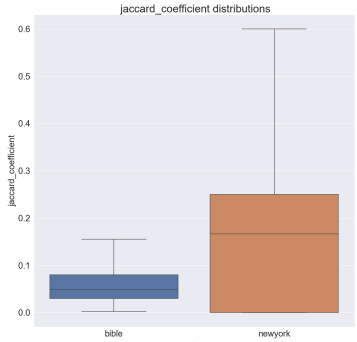
(b) Jaccard coefficient of **good** performing networks.



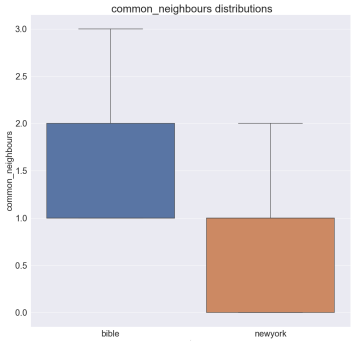
(c) Number of commmon neighbours of **good** performing networks.



(d) Adamic-Adar of **bad** performing networks.



(e) Jaccard coefficient of **bad** performing networks.



(f) Number of commmon neighbours of **bad** performing networks.

Figure 16: Network categories distributions.

We can see that when it comes to the Adamic-Adar index, the distributions are more similar in the pair of networks that perform better, with an almost identical median. When it comes to the worse performing pair, the distributions are disparate. As for the other features, there is no indication that the Jaccard coefficient can be relevant, since in both cases the distributions are different. In the case of common neighbours, the distributions in the better performing pair seem more comparable.

These findings can be an indication that a similar Adamic-Adar index distribution can lead to better performance. Nonetheless, this would not be an ideal measure to take into consideration before training or testing a classifier, since there is already some computation required to get the necessary pairs of nodes and calculate their Adamic-Adar index.

6 Conclusion

This study aimed to investigate transfer learning for link prediction. Specifically, we set to find whether it was possible for a link prediction model to generate good performance when tested on a different dataset. Moreover, we set to understand the underlying network characteristics that result in better or worse transfer learning performance.

To do so, we introduced a technique to perform cross-validation training across a wide sample of datasets. Through our experiments, we observed that transfer learning for link prediction is possible and that some network categories perform better as source for training and others to predict missing links on. The goal of this research was not to test the most state-of-the-art machine learning algorithms, but merely to test the possibility of transfer learning, which we successfully achieved. We did so by employing a simple classifier with default parameters, as well as standard and limited predictors for the node-pair based data input into the predictive model.

Furthermore, we found that clustering statistics are important indicators when picking a network to be used for training a predictive model. Namely, a high number of triangles per edge, or high clustering coefficient. When comparing the networks used for training and testing, the dissimilarity of their structural properties could also play an important role in the performance. Specifically, we found that when two networks show very dissimilar topologies, it is likely that the performance of transfer learning is hindered.

In future work, we would like to explore more advanced machine learning techniques such as automated machine learning to optimise the parameters of the classifiers. Additionally, we could add more features to the input dataset of the algorithm.

Lastly, the recent advancements in artificial intelligence prove that there are huge advancements being made that allow for pre-trained models to revolutionise entire industries. With this knowledge, researchers across all domains of network science could build upon our research to solve complex machine learning problems in networks.

References

[1] Adamic, L. A. and Adar, E. Friends and neighbors on the web. *Social networks*, 25(3): 211–230, 2003.

[2] Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[3] Barabási, A.-L. *Linked: The new science of networks*. American Association of Physics Teachers, 2003.

[4] Barabási, A.-L. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987):20120375, 2013.

[5] Barabási, A.-L. The hidden networks of everything, 2023. URL `https://www.youtube.com/watch?v=RfgjHoVCZwU`.

[6] Bender, E. A. and Williamson, S. G. *Lists, decisions and graphs*. S. Gill Williamson, 2010.

[7] Biggs, N., Lloyd, E. K., and Wilson, R. J. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[8] Bollobás, B. *Modern graph theory*, volume 184. Springer Science & Business Media, 1998.

[9] Bors, P. P. Topology-aware network feature selection in link prediction, 2022.

[10] Bradley, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[11] Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. Generative pretraining from pixels. In *International conference on machine learning*, pp. 1691–1703. PMLR, 2020.

[12] Daud, N. N., Ab Hamid, S. H., Saadoon, M., Sahran, F., and Anuar, N. B. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, 166:102716, 2020.

[13] de Bruin, G. J., Veenman, C. J., van den Herik, H. J., and Takes, F. W. Supervised temporal link prediction in large-scale real-world networks. *Social Network Analysis and Mining*, 11(1):1–16, 2021.

[14] Erdős, P., Rényi, A., et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[15] Euler, L. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pp. 128–140, 1741.

[16] Ghasemian, A., Hosseinmardi, H., Galstyan, A., Airoldi, E. M., and Clauset, A. Stacking models for nearly optimal link prediction in complex networks. *Proceedings of the National Academy of Sciences*, 117(38):23393–23400, 2020.

[17] Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[18] Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., and Azim, M. A. Transfer learning: a friendly introduction. *Journal of Big Data*, 9(1):102, 2022.

[19] Kunegis, J. and Preusse, J. Fairness on the web: Alternatives to the power law. In *Proceedings of the 4th Annual ACM Web Science Conference*, pp. 175–184, 2012.

[20] Kunegis, J., Staab, S., and Dünker, D. KONECT – The Koblenz Network Collection. In *Proc. Int. Sch. and Conf. on Netw. Sci.*, 2012.

[21] Kuo, T.-T., Yan, R., Huang, Y.-Y., Kung, P.-H., and Lin, S.-D. Unsupervised link prediction using aggregative statistics on heterogeneous social networks. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 775–783, 2013.

[22] Leskovec, J., Huttenlocher, D., and Kleinberg, J. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pp. 641–650, 2010.

[23] Liben-Nowell, D. and Kleinberg, J. The link prediction problem for social networks. In *Proceedings of the 12th International Conference on Information and knowledge management*, pp. 556–559, 2003.

[24] Lichtenwalter, R. N., Lussier, J. T., and Chawla, N. V. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 243–252, 2010.

[25] Lü, L. and Zhou, T. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[26] Menon, A. K. and Elkan, C. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases: European Conference*, pp. 437–452. Springer, 2011.

[27] Mignone, P. and Pio, G. Positive unlabeled link prediction via transfer learning for gene network reconstruction. In *Foundations of Intelligent Systems: 24th International Symposium*, pp. 13–23. Springer, 2018.

[28] Muniz, C. P., Goldschmidt, R., and Choren, R. Combining contextual, temporal and topological information for unsupervised link prediction in social networks. *Knowledge-Based Systems*, 156:129–137, 2018.

[29] Narkhede, S. Understanding auc-roc curve. *Towards Data Science*, 26(1):220–227, 2018.

[30] Newman, M., Barabási, A.-L., and Watts, D. J. *The Structure and Dynamics of Networks*. Princeton University Press, Princeton, 2006.

[31] Newman, M. E. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

[32] Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[33] OpenAI. Gpt-4 technical report. 2023. URL `https://cdn.openai.com/papers/gpt-4.pdf`.

[34] Papadimitriou, A., Symeonidis, P., and Manolopoulos, Y. Fast and accurate link prediction in social networking systems. *Journal of Systems and Software*, 85(9):2119–2132, 2012. Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011).

[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[36] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

[37] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL `https://arxiv.org/abs/2103.00020`.

[38] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents, 2022.

[39] Schlender, T. An empirical investigation of an accredited supervised link prediction method, 2021.

[40] Torrey, L. and Shavlik, J. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264. IGI global, 2010.

[41] Tsung, F., Zhang, K., Cheng, L., and Song, Z. Statistical transfer learning: A review and some extensions to statistical process control. *Quality Engineering*, 30(1):115–128, 2018.

[42] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł.,

and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[43] Wang, P., Xu, B., Wu, Y., and Zhou, X. Link prediction in social networks: the state-of-the-art. *CoRR*, abs/1411.5118, 2014. URL http://arxiv.org/abs/1411.5118.

[44] WEI, Y., Zhang, Y., Huang, J., and Yang, Q. Transfer learning via learning to transfer. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5085–5094. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/wei18a.html.

[45] Wikipedia, the free encyclopedia. Confusion matrix, 2021. URL https://en.wikipedia.org/wiki/Confusion_matrix. [Online; accessed April 2023].

[46] Wikipedia, the free encyclopedia. Economics gini coefficient2, 2021. URL https://en.wikipedia.org/wiki/File:Economics_Gini_coefficient2.svg. [Online; accessed April 2023].

[47] Zhu, C. Chapter 6 - pretrained language models. In Zhu, C. (ed.), *Machine Reading Comprehension*, pp. 113–133. Elsevier, 2021.

A Additional Material

Network	Category	Number of Nodes	Number of Edges	Max Degree	Average Degree	Diameter	90% Effective Diameter	Median Distance	Mean Distance	Gini Coefficient	Degree Assortativity	Triangle Count	Clustering Coefficient	Transitivity
adolescent	Human social	2539	10455	27	8.2355	8	5.3048	5	4.5165	0.2899	0.2513	4694	0.1467	0.1419
airtraffic	Infrastructure	1226	2408	34	3.9282	17	8.0542	6	6.0979	0.3987	-0.0152	326	0.0675	0.0639
asoif	Miscellaneous	796	2822	122	7.0905	9	4.1437	3	3.4108	0.6094	-0.1156	5651	0.4858	0.209
astroph	Co-authorship	17903	196972	504	22.0044	14	5.0028	4	4.1743	0.5992	0.2013	1350014	0.6328	0.3178
astrophysics	Co-authorship	14844	119651	360	16.1211	14	7.0119	5	5.1085	0.5767	0.2277	754159	0.6697	0.4253
bible	Lexical	1707	9059	364	10.6139	8	3.9365	3	3.3763	0.5531	-0.0519	19936	0.71	0.1625
blogs	Hyperlink	1222	16714	351	27.3552	7	3.2894	3	2.7222	0.622	-0.2213	101043	0.3203	0.226
caida	Computer	26475	53381	2628	4.0326	17	4.637	4	3.9125	0.6281	-0.1946	36365	0.2082	0.0073
chesapeake	Trophic	39	170	33	8.7179	3	1.9552	2	1.8286	0.3175	-0.3758	194	0.4502	0.2842
chess	Interaction	7115	55778	181	15.679	13	4.831	4	3.9499	0.607	0.3705	108580	0.1794	0.1258
congress	Interaction	219	521	33	4.758	7	3.8707	3	3.1837	0.5255	-0.3395	212	0.2553	0.1191
contiguous	Infrastructure	49	107	8	4.3673	11	6.9838	4	4.255	0.2012	0.2334	57	0.4967	0.4062
copperfield	Lexical	112	425	49	7.5893	5	2.9806	2	2.4724	0.4173	-0.1293	284	0.1728	0.1569
cora	Citation	23166	89157	377	7.6972	20	6.9523	6	5.7382	0.5202	-0.0553	78791	0.266	0.1169
dblp	Citation	12494	49578	713	7.9363	9	4.9979	4	4.3722	0.6563	-0.046	43896	0.1181	0.062
digg	Communication	29652	84780	283	5.7183	12	5.4019	5	4.6804	0.626	0.0026	4282	0.0054	0.0056
dnc	Communication	1833	4366	402	4.7638	8	3.9822	3	3.379	0.7166	-0.3088	9431	0.2157	0.089
dolphins	Animal	62	158	12	5.0968	8	5.0523	3	3.4543	0.3283	-0.0549	94	0.2525	0.3079
eat	Lexical	23132	297094	1062	25.6868	5	3.8524	3	3.4314	0.6739	-0.0477	409174	0.0888	0.0404
erdos	Co-authorship	6927	11849	507	3.4211	4	3.8764	4	3.7914	0.646	-0.1156	5969	0.1239	0.0357
euroroad	Infrastructure	1039	1305	10	2.512	62	33.3405	17	19.1812	0.2332	0.09	32	0.0189	0.0353
facebook	Communication	43953	182384	223	8.299	18	6.8357	6	5.7112	0.584	0.216	122842	0.1149	0.0851
filmtrust	Online social network	610	1119	67	3.6689	13	6.8845	5	5.0101	0.5247	0.017	725	0.1766	0.1883
florida_dry	Trophic	128	2106	110	32.9062	3	1.8775	2	1.7223	0.2506	-0.1044	8715	0.3346	0.3143
florida_wet	Trophic	128	2075	110	32.4219	3	1.878	2	1.7259	0.2527	-0.1117	8437	0.3346	0.3119
foldoc	Hyperlink	13356	91470	728	13.6972	7	4.5814	4	3.9909	0.307	-0.0122	104221	0.3379	0.1132
football	Interaction	115	613	12	10.6609	4	2.8281	3	2.3968	0.04	0.1624	810	0.4032	0.4072
gnutella25	Computer	22663	54693	66	4.8266	11	6.5485	6	5.6154	0.5417	-0.1734	806	0.0053	0.0045
hamsters	Online social network	2000	16097	273	16.097	10	4.7972	4	3.6685	0.5488	0.0227	52664	0.5401	0.2295
hepph	Citation	34401	420784	846	24.4635	13	5.2299	4	4.3953	0.5235	-0.0064	1276859	0.2856	0.1457
hepth	Citation	27400	352021	2468	25.695	15	5.3557	4	4.2696	0.5658	-0.0305	1478698	0.3139	0.1196
karate	Human contact	34	78	17	4.5882	5	3.44	2	2.4433	0.3854	-0.4756	45	0.5706	0.2557
lesmis	Miscellaneous	77	254	36	6.5974	5	3.3951	3	2.6356	0.461	-0.1652	467	0.5731	0.4989
littlerocklake	Trophic	183	2433	105	26.5902	4	2.6487	2	2.1278	0.42	-0.2649	11292	0.3239	0.3325
networkscience	Co-authorship	379	914	34	4.8232	17	9.1821	6	6.2853	0.3734	-0.0817	921	0.7412	0.4306
newyork	Infrastructure	264346	365050	8	2.7619	720	424.937	258	261.456	0.1891	0.1785	6529	0.0208	0.0254
openflights	Infrastructure	3397	19230	248	11.3218	13	5.3951	4	4.193	0.7155	-0.0078	101112	0.4883	0.2483
pgp	Online contact	10680	24316	205	4.5536	24	10.0699	8	7.6529	0.5918	0.2382	54788	0.2659	0.378
proteins	Metabolic	1615	3106	95	3.8464	13	6.9724	5	5.0931	0.5777	-0.202	96	0.0063	0.0058
reactome	Metabolic	5973	145778	855	48.8123	24	5.3904	4	4.142	0.6473	0.2414	4187395	0.6091	0.6055
residence	Human contact	217	1839	56	16.9493	4	2.7857	2	2.3275	0.2456	0.096	3629	0.3628	0.3036
routeviews	Computer	6474	12572	1458	3.8838	9	4.4492	4	3.6669	0.6069	-0.1818	6584	0.2522	0.0096
sistercities	Miscellaneous	10320	17987	99	3.4859	25	10.1647	8	7.6539	0.5167	0.3515	6261	0.0478	0.1114
slashdot	Communication	51083	116573	2915	4.5641	17	5.2801	5	4.5891	0.6826	-0.0347	18937	0.0201	0.006
twitter	Online social network	22322	31823	238	2.8513	14	7.5594	6	6.1882	0.6147	-0.4901	8784	0.0715	0.0216
uc_irvine	Communication	1893	13835	255	14.617	7	3.6579	3	3.069	0.6519	-0.188	14319	0.1097	0.0568
wikipedia	Online contact	7066	100667	1065	28.4933	7	3.7854	3	3.2495	0.7463	-0.0833	607279	0.1417	0.1254
wordnet	Lexical	145145	656230	1008	9.0424	16	6.3189	5	5.362	0.5502	-0.0625	1144648	0.6035	0.0958
yeast	Metabolic	1458	1948	56	2.6722	18	9.5891	7	7.0697	0.4626	-0.2095	206	0.0708	0.0518

Table 6: Network Topologies