Comparison of Hellinger Distance and C4.5 Decision Trees for the Class Imbalance Problem of Link Prediction

Social Network Analysis for Computer Scientists — Course Project Paper

Sevak Mardirosian
LIACS, Leiden University
s.mardirosian@umail.leidenuniv.nl

Michail Tsiaousis
LIACS, Leiden University
m.tsiaousis@umail.leidenuniv.nl

ABSTRACT

In a group or a community where people interact and form associations with each other, it usually holds that the number of people that are connected is dramatically less than those that do not. Representing such a community as a social network, the aim is to predict those pairs of nodes that will connect in the future. This is known as the *Link Prediction* problem. Utilizing supervised learning, the latter can be phrased as binary classification. In this scheme, pairs of nodes that are connected represent the positive class, whereas those that do not correspond to the negative. Hence, there is a great imbalance between the two classes since the number of links that do not exist is overwhelmingly greater than those that do. Classification with imbalanced data can pose a problem because many algorithms tend to classify every observation in the majority class. In this paper, we apply Hellinger Distance Decision Trees (HDDT), which are considered to be skew insensitive and well suited for imbalanced data, to seven data sets constructed from five social networks. We compare HDDT to C4.5 which is the standard choice of decision trees in the machine learning community. We evaluate our results using the Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curve.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning; E [**Data**]: Miscellaneous

General Terms

Theory; Experimentation

Keywords

Social Networks; Link Prediction; Decision Trees; Learning; Hellinger Distance Decision Trees; C4.5

1. INTRODUCTION

Interactions between people can be visualized and modeled using social networks. Each node represents a person, and an edge between two nodes corresponds to some sort of association. These connections are usually formed because of common interests between the two persons. Since social networks dynamically change over time, new connections form and others cease to exist. An interesting problem to examine is whether or not it is possible to predict the formation of new links, that is, predict the state of the network in some future time. In this paper the problem we are concerned with is the prediction of a future link between two nodes in the graph. This is also known as the *Link Prediction* problem.

More formally, the link prediction task can be formulated as follows. Given a network at time t, accurately predict the links that are going to form at a future time t'.

Some examples of link prediction can be found in organizations such as Amazon, where it is possible to extract information based on raw data in order to predict what the customer might actually buy or find interesting. Another example would be the professional social-media platform, Linked-in. Based on data they would be able to, for instance, predict your next connection; link back jobs which are relevant to you; link contents/articles based on your connections, interests, and things you read on the web. Similarly, security corporates could more precisely focus their efforts based on probable relationships in malicious networks, and researchers can easily adapt link prediction methods to identify links that are surprising given their surrounding network, or links that may not exist at all [14].

In this paper, we approach the problem of link prediction using supervised learning [8, 10, 14, 22]. Unsupervised techniques have also been used. In both cases, we calculate metrics, also called features or baseline predictors, between pairs of nodes that indicate how likely it is for the pairs to connect in the future. Those metrics can be of topological nature such as the number of common neighbors between two nodes, or domain specific, such as the number of papers that two authors have written together in the case of a collaboration network. The difference between supervised and unsupervised learning is that in supervised learning the output datasets are provided which are used to train the machine and get the desired outputs whereas in unsupervised learning no datasets are provided, instead the data is clustered into different classes .

In the framework of supervised learning, the prediction task is framed as a binary classification problem where a classifier is built in order to distinguish between the instances of the positive and negative class. We say that a pair of nodes belong to the positive class if they are connected, and to the negative class otherwise. One of the main problems in link prediction is the class imbalance, that is the number of instances of the negative class are overwhelmingly greater than those of the positive. This means that the number of links that do actually form (positive class) is dramatically less than the number of links that do not form (negative class) [14]. The reason that class imbalance consists a problem is due to the fact that it impedes the ability of classifiers to learn from the minority class [2], resulting to a classifier that predicts every instance as one belonging to the majority class.

A method for dealing with imbalanced data sets is that of applying a specific type of decision tree called Hellinger Distance Decision Trees (HDDT) [1, 2], which are robust and insensitive to skewed distributions. This method is suggested in [14] as a solution to solve the problem of class imbalance in link prediction, but the authors have not provided results of this method. An extensive comparison of HDDT and C4.5 can be found in [2]. In this latter study, HDDT outperformed C4.5 in most imbalanced data sets for binary classification, whereas C4.5 performed better in the multiclass case. The differences in results between the two types of trees were not substantial with respect to the evaluation metrics that the authors used. To the best of our knowledge, HDDT have not been applied in the link prediction domain. In this paper we compare C4.5 decision trees [19] which have been applied in [10], and HDDT as proposed in [14]. Our interest is to examine the performance of HDDT and C4.5 for the class imbalance problem of link prediction. A discussion of C4.5 and HDDT can be found in Section 4.

The paper is structured as follows. In Section 2, we formalize the link prediction problem. Specifically, we discuss the class imbalance problem, and the transformation of a network to a tabular data set. Related work is provided in Section 3. In Section 4, we describe the approaches that have been taken in [10] and [14], and we also provide an analysis of HDDT and C4.5. Section 5 describes the data sets and software in which the experiments and results of Section 6 are based on. Finally, we conclude in Section 7.

2. PROBLEM STATEMENT

2.1 Preliminaries

Consider at a particular time t, an (un)directed network $G = \langle V, E \rangle$, where V is the set of nodes and E is the set of edges. The goal is to predict the edges that will form in the network at time $t' > t$. As an example, consider a collaboration network of authors. Two authors v and w are linked when they have published at least one research paper together. The goal is to predict future connections between authors that have not yet collaborated. This paper concerns undirected networks since the software we are using supports only this type of networks. All directed networks that we consider are first converted to undirected. Regarding the data sources and software used in our experiments, we refer the reader to Section 5.

In this paper, we approach the problem of link prediction using supervised learning. In this framework, in order to build models with predictive ability, one needs to construct two sets, mainly a *train* and a *test* set. Then, the model learns features and patters from the train set, and it is evaluated on the test set. These two sets form together the *data set.* This set is of the form $(\vec{x}, y)$, where $\vec{x}$ indicates the inputs to the model, and y indicates the label, that is whether or not a link is present between two nodes. The model learns from the instances $(\vec{x}, y)$ of the train set. In the evaluation phase, we feed the instances $\vec{x}$ of the test set to the trained model, and the output is the predicted label, i.e our prediction of whether or not a link will form between two nodes.

2.2 Constructing Data Sets

The process of constructing the data set is described in detail in [10, 14]. In order to transform the network to a data set, we first choose two adjacent periods, the *train* and *test period.* In the train period, denoted by $[0, t_k]$, we have pairs of nodes that are connected, that is, they have some sort of association or connection. The same holds for the test period $[t_{k+1}, t_l]$. We construct a new network $G_k = \langle V_k, E_k \rangle$ consisted of all pair of nodes that do not share a connection in the train period. We then examine whether or not they are connected in the test period. If they do, we assign to the pair a positive label, and otherwise a negative label. The data set contains $\binom{|V_k|}{2} - |E_k|$ instances, since we do not account for self-loops, and in undirected networks each pair is recorded once.

Selecting an appropriate set of features is one of the most important steps in supervised learning algorithms [10]. The inputs to the model are selected by calculating topology-based and domain specific metrics from G_k that will serve as attributes to the data set. Such metrics are, but not limited to, *common neighbors*, *Jaccard coefficient*, *Adamic Adar*, *preferential attachment*, *Katz* measure, *Rooted PageRank*, and many others. 6 provides an explanation of the metrics used in our experiments. In the supervised case, we use state-of-the-art predictive models such as Neural Networks, Support Vector Machines, Decision Trees, and Random Forests that treat these individual metrics as inputs. The output or prediction of the model, is whether or not a link will form between two specific nodes.

That being said, we end up with a data set in the standard format $(\vec{x}, y)$, where $\vec{x}$ are metrics extracted from G_k and y indicates the label, that is whether or not a pair of nodes is connected or not.

2.3 Class Imbalance and Size

Although we have constructed the data set, there are two main problems. The first is that the size of the data set is very large which can make the training process infeasible. For example, consider the relatively small in size `MathOverflow` network [17] of the SNAP repository [1], consisted of 21688 nodes and 107581 edges. Defining the train period to be two-thirds of the networks' edges, and assuming that all node appear in V_k, the resulting data set would be of size $\binom{21688}{2} - 107581$, which yields more than 235 million instances.

[1]`http://snap.stanford.edu/index.html`

As we mentioned in Section 1, one of the main problems in link prediction is the imbalance between the positive and negative class. That is, in the constructed data set, the amount of instances with a negative label is overwhelmingly greater than those with a positive label.

A technique called under-sampling provides a solution to the class imbalance problem. In under-sampling, we acquire a representative sample from the majority class that is more or less equal to the size of the minority class. In this way, we create a balanced data set comprised from the sample of the majority class and all instances of the minority class. We then say that the data set is under-sampled to balance. Note that, when a data set is under-sampled to balance it means that the ratio of positive and negative instances is 1:1. It is possible to under-sample the data set at different ratios. To see this, consider the case where the class imbalance is of size 1:5000 and the data set has 5 million instances. Training a model with this setup is a difficult task due to the huge size of the data set. In this case, it would be helpful to under-sample the data set to a ratio different than 1:1, for example to a ratio of 1:10. That is, we explicitly acquire a sample of the majority class that is ten times larger than that of minority's class. In this way, we decrease the size of the data set and we retain imbalanced classes.

A second method that mitigates both problems of size and class imbalance is proposed in [14]. As we mentioned in Section 2.2, in the process of constructing the data set we consider each pair of nodes that do not exist in the train period as one that could potentially connect in the future. It is suggested to treat each neighborhood as a separate problem. We define the n-neighborhood of a node v as the set of nodes that are 0 to n steps away from v. Instead of considering all pairs of nodes to enter the data set, we choose for each node, and for a specified n, only those that belong to the n-neighborhood of the node. In this way, both the size of the data set and the imbalance between the classes decrease greatly. For the case of `MathOverflow` network, choosing $n = 2$ results to a data set with more than 5 million instances of which only 3168 belong to the positive class, an imbalance of more than 1:1578.

3. RELATED WORK

The link prediction problem was formalized in [12], where there is an extensive analysis of different baseline predictors, applied to co-authorship networks. Nevertheless, state-of-the-art supervised learning models are not studied in this paper. Furthermore, co-authorship networks have been analyzed in [10], [18], and [4], but this time several machine learning algorithms such as Decision Trees, SVM, Neural Networks, and K-Nearest Neighbors are used for prediction. In contrast to [10], where static, unweighted networks are used, [18] takes into account the evolution of the network, and the authors of [4] analyze a weighted network. Several recommendations and suggestions are presented in [14], regarding how to perform link prediction, and how to approach inherent issues such as dealing with class imbalance, variance reduction, and huge train sets.

In the domain of counter-terrorism, link predictions has been studied in [5], [8]. In the latter paper, several networks are constructed by removing nodes from the original graph, called visible networks. A set of features is calculated from these networks in order to predict links of the original one, using supervised learning models. A more probabilistic approach is taken in [6], where chance-constrain programs are used, and in [21], where the framework of Relational Markov Network is utilized. Finally, [23], and [11] are extensive surveys regarding the different baseline predictors, classification methods, and approaches that have been used in the link prediction domain.

4. SUGGESTED APPROACHES

In this section, we describe the approaches that have been taken in [10] and [14], and we discuss the differences between them. In Subsection 4.4 we give a description of HDDT and C4.5.

4.1 Construction of Data Sets

The authors in [10] use two undirected, unweighted collaboration networks of authors, specifically the BIOBASE [2] and DBLP [3] networks. An edge between two authors/nodes exists in case they have written at least on research paper together. Both networks contain timestamps of the years of collaborations. For each network, the authors choose a corresponding train and test period that span one or more years. For example, in the first network, the train period is consisted of 5 years from 1998 to 2002, whereas the test period comprises the year 2003. The data set constructed from this network contains each combination of authors that do not exist in the train period, that is, each pair that could potentially have a connection in the future. For each of these combinations, it is examined whether or not there is a connection in the test period. If a connection exists, a positive label is assigned to the pair, otherwise a negative label.

In [14], the authors use two networks, one weighted, undirected co-authorship network of 19464 edges of condensed matter physics collaborations from 1995 to 2000 called `condmat`, and one directed weighted network of 712 million cellular phone calls, called `phone`. The process of constructing the data sets is restricted in the n-neighborhood of each node. Specifically, instead of considering all possible combinations of pairs of nodes that do not exist in the train period, the data set is constructed only from those combinations that are not part of the train period, but also contained in the neighborhood of n size. Specifically, for each node v, the n-size neighborhood is calculated. We then compute the metrics between v and the nodes of this neighborhood by excluding pairs that already belong to the train period. The idea behind this is that, it is more possible for links to form in the future between nodes that are closely together, than between nodes that are far apart. Thus, reducing the problem to each neighborhood offers two advantages. Firstly, the imbalance between classes is decreased, and secondly given that specific metrics such as *Adamic Adar* and *Rooted PageRank* are expensive to compute, we avoid calculating those metrics for pairs of nodes that are not highly likely to connect in the future. The value of n is chosen by the user. Selecting higher values guarantee that we do not lose connections that actually do form, but on the other hand the

[2]`http://www.elsevier.com`
[3]`http://dblp.uni-trier.de/xml/`

instances of the negative class increase dramatically, along with the computation time of the metrics.

4.2 Calculation of Features

The authors in [10] calculate several topological and domain specific features. Topological features include the *shortest dictance* and *clustering index*, whereas the domain specific features include the sum of papers the two authors share, the sum of the count of keywords in the papers of each author, the number of common keywords two authors have used etc.

On the other hand, the authors of [14] concentrate on features that can be calculated for every network. The High Performance Link Prediction (HPLP) framework is presented which comprises several topological metrics such as *In/Out Degree*, *Common Neighbors*, *Maximum Flow*, *Adamic Adar*, *Jaccard Coefficient*, and others. Note that this is different from the work of Hasan et al. [10] where most of the selected features are domain specific. Hence, HPLP serves as a general framework for link prediction. Additionally, a new unsupervised prediction method is presented called `PropFlow`. It is a predictor based on random walk that starts from node v_i and ends at a node v_j in l or fewer steps, using link weights as transition probabilities [14]. A score s_{ij} is produced for each pair of nodes v_i and v_j that serve as an estimation of the likelihood that the two nodes will be connected in the future. `PropFlow` is also included in HPLP.

4.3 Overcoming Imbalance

In [10], the problems of huge data sets and that of class imbalance are solved by under sampling the data set to balance. After creating a balanced data set, the authors train several machine learning models such as Decision Trees (with and without bagging), Support Vector Machines, Naive Bayes, Neural Networks and others. They found that Decision Trees with bagging, and Support Vector Machines provided the best performances when compared to different metrics such as accuracy, precision, recall, F-value, and squared error. The models are evaluated using 5-fold cross validation.

Under-sampling to balance is also used in [14]. The authors use the features proposed in the HPLP framework and apply Random Forests with bagging, for different values of the neighborhood size n. They found that with increasing values of n there is a deterioration of performance in both networks. The value of $n = 2$ provided the optimal results.

4.4 HDDT and C4.5

4.4.1 Preliminaries

In this sub-section we give a brief introduction to decision trees and we discuss the different ways in which C4.5 and HDDT operate. The only difference between the two is the way in which they are constructed.

Classification trees are tree-like structures that consist of nodes that can be either a root node, internal nodes, or terminal nodes, also called leaves. The nodes are connected with each other from top to bottom with edges, also called branches. The root node is the first node of the tree, whereas the leaves are the final nodes. For each node, there are two outgoing branches, the *left* and *right* branch that lead to the next two nodes. The left and right branches are called

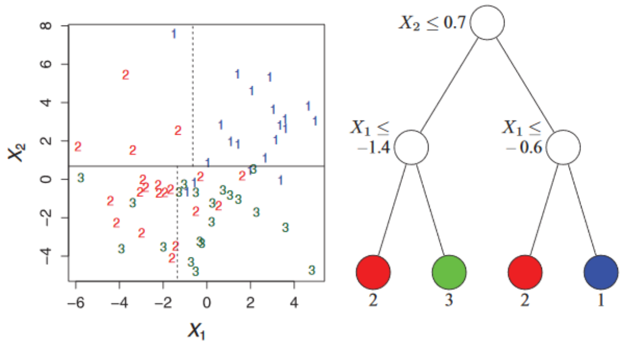

Figure 1: Left: Recursive partitioning of the 2-dimensional space formed by X_1, X_2. Right: The resulting decision tree.

child nodes and the node from which they resulted is called the *parent* node. In order to classify an instance to one of the two classes, in each internal node, we test whether or not a specific condition is satisfied. If it is, we follow the left branch, and otherwise the right one. By evaluating over those conditions we arrive at the leaves that indicate the class prediction for a particular observation of our data set.

4.4.2 General Construction - Recursive Partitioning

Consider z observations of p features $X_1, X_2, ..., X_p$, and a class variable Y that indicates the class for each instance. In our case, z is the number of instances in the data set, X_i, $i = 1, 2, ..., p$ indicate the calculated features from network G_k, and variable Y corresponds to the class, that is whether or not the instance has formed in the test period. Variables X_i, $i = 1, 2, ..., p$ form a p dimensional space in which the observations lie. Decision trees recursively partition this space into non-overlapping rectangles. The partitioning is accomplished in the following way. One of the X_i variables is selected, say X_r, and splits the p-dimensional rectangle into two parts. One part contains all observations for which $X_r \leqslant s_r$, and the second part contains all observations for which $X_r > s_r$, where s_r is a value of feature X_r. The two remaining parts are partitioned in the same fashion by either the same variable or by an other $X_i, i = 1, 2, ..., p,\ p \neq r$. The goal is to split this hyper dimensional space in the best possible way, that is partition the space such as each rectangle contains observations only from a specific class.

Figure 1 taken from [15], illustrates the partitioning of the 2-dimensional space with features X_1, X_2 on the left and the corresponding decision tree on the right. The tree has two internal nodes and four leaves. The condition on the root node tests for each instance of the data set, whether or not $X_2 \leqslant 0.7$. If the condition holds we follow the left branch where we find the second condition, this time with respect to X_1. If $X_1 \leqslant -1.4$ holds, we predict that the instance belongs to the red class, otherwise to the green class.

4.4.3 Splitting Criteria - Gain Ratio

An important question that we could ask regarding the construction of decision trees is how do we choose the features X_i, $i = 1, 2, ..., p$ in order to partition the p-dimensional space. It is reasonable that some features will provide better splits than others. The aim is to choose the feature that splits the parent node into two child nodes that are homogeneous or "pure", that is, each child node contains instances of only one class. This means that the feature we select is able to discriminate well between the observations of the two classes. Each type of tree uses a different splitting criterion. C4.5 decision trees use the *Gain Ratio* which is based on the calculation of entropy, whereas HDDT rely on *Hellinger Distance*.

The gain ratio, also called information gain, measures the amount of information we gain by splitting the data using a specific feature. Features that result into "pure" or homogeneous splits yield higher values of gain ratio. Essentially, by increasing the gain ratio we achieve reduction in entropy. Entropy measures the impurity of an arbitrary collection of examples. It takes values in the interval $[0, 1]$. High uncertainty corresponds to non-homogeneous splits in the two child nodes since we do not discriminate well between the two classes, which results to higher values of entropy. In formal notation, the entropy of a set S is given by

$$I(S) = -\sum_{b=1}^{T} p_b \ log_2(p_b)$$

where T is the number of classes ($T = 2$ for binary classification), and p_b, $b = 1, 2, ..., T$ indicates the proportion of instances in class b. Using a specific feature X_i for splitting the p-dimensional space, by calculating the entropy of the parent node and the expected entropy of the resulted left and right child nodes, the gain ratio is calculated as

$$G(feature) = I(S) - \sum_{v} \frac{|S_v|}{|S|} I(S_v)$$

where S indicates the parent node and S_v the child node. The result of gain ratio indicate the information we have gained by splitting the p-dimensional space using the selected feature. Using a greedy search, the feature that yields the highest gain ratio is chosen for the split.

4.4.4 Splitting Criteria - Hellinger Distance

HDDT use the Hellinger Distance as a splitting criterion for the construction of the tree. Hellinger distance is a measure of distributional divergence [1]. Let P and Q be two distributions of normalized frequencies of feature values across classes. If there is no distinction between P and Q, that is, if P and Q overlap then the hellinger distance between the two distributions is equal to 0. In case P and Q are completely disjoint, the distance takes its maximum value, $\sqrt{2}$. The hellinger distance is utilized as a splitting criterion in the following way. Consider two class distributions X_+ and X_- containing the instances of the positive and negative class, respectively. For a specific feature X_i, $i = 1, 2, ..., p$, the hellinger distance is given by

$$d_H(X_+, X_-) = \sqrt{\sum_{j=1}^{p} \sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}}}$$

where j indicates the number of partitioned rectangles. For binary classification $j = 1, 2$. X_{+j} indicate the instances of the positive class in rectangle j, whereas X_+ represents all positive instances. The same holds for X_{-j} and X_- for the negative class. Due to the fact that the number of instances of the positive (minority) class in rectangle j is conditioned with respect to all positive instances, and because the prior probabilities do not explicitly appear in the distance calculation, hellinger distance is considered to be skew insensitive and well suited for classification in imbalanced data. A prior probability $P(+)$ is the probability of a randomly chosen instance to belong in the positive class. In the case of imbalanced data where the number of positive instances is less than the number of negatives, the prior probability $P(+)$ is very low. Hence, the fact that hellinger distance conditions the number of positive instances of rectangle j to all positive instances of the data set without taking into account the prior probabilities, makes HDDT robust and well suited for classification in imbalanced data. The distance $d_H(X_+, X_-)$ is calculated for each feature. Using a greedy search, we choose the feature that returns the maximum distance.

5. DATA SETS AND SOFTWARE TOOLS

5.1 Data Sources

Almost all link prediction works need to verify their methods on the collected datasets. The datasets are important for fairly reproducing and comparing different link prediction methods. Constructing and collecting the datasets is a time-consuming and labor-intensive work. That said, not all the datasets are publicly available to use and some of them are incomplete. Since the process of transforming the network to a data set requires the creation of two non-overlapping periods that simulate the formation of links between nodes from one period to the other, it is essential to consider networks that contain timestamps. We are interested in both directed and undirected networks. Nevertheless, we convert the former type to the latter. In all of our experiments, we consider unweighted networks.

A summary of the size of nodes and edges, along with the length of the timespan of the social networks we consider can be found in Table 1. We experiment with networks of different sizes and timespans in order to examine how the results differ for each case. We use three networks that come from the KONECT [4] database. The first one is `UCIrvine` [16], a directed network containing sent messages between the users of an on-line community of students from the University of California, Irvine. Next, `Digg` [3] is a reply directed network of the social news website Digg where each node in the network is a user of the website, and each directed edge denotes that a user replied to another user. Furthermore, we use `Slashdot` [9], a directed reply network of the technology website Slashdot, where nodes and edges represent users and replies, respectively. Moreover, we consider `RealityCall` [7, 20] from Networks Repository [5], which is an undirected network of mobile phone call events between a small set of core users at the Massachusetts Institute of Technology (MIT). The network also contains edges between users that do not belong in the small set of users, who called other individuals that were not actively monitored. Therefore, some sort of noise might exist in this network, since we do not have more information about the individuals outside of the small set of MIT users. Finally, we consider the directed `MathOverflow` network [17] that can be found on SNAP repository. A node

[4]`http://konect.uni-koblenz.de/networks/`
[5]`http://networkrepository.com/ia-reality-call.php`

	UC	Digg	Slashdot	Reality	MathOver
Nodes	1899	30398	51083	6809	21688
Edges	59835	87627	140778	7680	107581
Timespan	6 months	16 days	2 years	4 months	6.5 years

Table 1: Number of nodes, edges, and length of timespan for the 5 social networks.

represents a user, and an edge indicates that user u answered user's v question on the website. All five networks are well suited for the link prediction problem. This is due to the fact that the nature of these networks is the interaction of individuals and we expect that connections taken place at time t form a base for the connections that will form in the near future. Hence, we learn the patterns from interactions at time $[0, t]$, in order to predict future connections at $t' > t$.

5.2 Software Tools

Although there are many link prediction metrics and methods proposed, only very few works open their source codes. Re-implementing methods and formulas to calculate predictors is a time-consuming process. Only few public tools try to integrate these metrics and methods such as `linkpred`[6] and `LPmade` [7] [13] which both have a handful of link prediction metrics. `LPmade` is a cross-platform software solution that provides multi-core link prediction and related tasks and analysis [13]. It is written in `C++` and therefore it is suited for handling very large networks. Unfortunately, compilation issues, and the lack of documentation and support prevented us from using the software. Hence, we used a `Python` library called `linkpred`. The main disadvantage of `linkpred` is that it is entirely written in `Python` and it can prove to be very slow for handling large networks.

6. EXPERIMENTS AND RESULTS

6.1 Experimental Setup

We performed experiments in five social networks. All directed networks are converted to undirected, due to the limitations of `linkpred` library. The train set is constructed by considering the n-neighborhood size, where $n = 2$. For each network, we apply HDDT and C4.5 using the parameters proposed in [2]. This setup consists of using bagging with 100 trees, un-collapsed, un-pruned, with Laplace smoothing at the leaves. Due to the huge size of the data sets, in order to be able to train decision trees but at the same time retain the imbalance, we under-sample the data sets to specified ratios. Specifically, we under-sample the larger networks `Digg, MathOverflow` and `SlashDot` to 1:10 ratio. Furthermore, we under-sample the smaller `UCIrvine` network to 1:10 and 1:60. From now on we will refer to them as `UCIrvine10` and `UCIrvine60`, respectively. Finally, due to the fact that `RealityCall` has only 56 positive instances, we can under-sample the data set to more extreme ratios. For `RealityCall` we use two versions, one under-sampled to 1:10 and the second to 1:600. We refer to them as `RealityCall10` and `RealityCall600`. In our experiments we also examine how the accuracy of the classifiers change when we move from smaller ratios of imbalance to larger ones. We perform 10-fold cross validation with stratified sampling for creating the train and test sets for each fold. A modified version of the WEKA software [24] is used that contains the implementation of HDDT, which can be found in `https://www3.nd.edu/~dial/software/`.

[6]`https://github.com/rafguns/linkpred`
[7]`https://github.com/rlichtenwalter/LPmade`

6.2 Evaluation

We evaluate the classifiers using the Receiver Operating Characteristic (ROC) curve and the Precision-Recall (PR) curve. The Area Under ROC (AUROC) metric is also provided, which quantifies the performance of a classifier. The main advantage of ROC and PR curve over single metrics such as accuracy is that they evaluate the classifier over all possible thresholds, whereas accuracy is specified only for a specific threshold.

ROC curves contain the True Positive Rate (TPR), also called Recall on the y-axis and the False Positive Rate (FPR) on the x-axis. Recall is given by $\frac{TP}{TP+FN}$, where TP and FN indicate the *True Positives* and *False Negatives*, respectively. Thus, Recall gives the probability of classifying correctly a positive instance, given that it is positive. The x-axis represents the FPR, given by $\frac{FP}{FP+TN}$, where FP indicates the *False Positives*, and TN the *True Negatives.* A ROC curve closer to the upper left corner indicate a classifier that can accurately predict positive instances given that they belong to the positive class, for a low misclassification rate in the negative class, that is low FPR. The classifier that predicts each instance correctly has an AUROC value of 1. A classifier that is not able to discriminate between classes has an AUROC of 0.5 and its curve is a diagonal line at $f(x) = x$. Using ROC curves and the AUROC metric we can compare two or more classifiers. The one that has a curve closer the the upper left corner and thus greater AUROC value is considered to be a better classifier.

A supplementary metric to the ROC curve is the PR curve. The y-axis represents the precision, given by $\frac{TP}{TP+FP}$, that is the proportion of correctly classified positive instances over all instances that were predicted as positive. In other words, precision yields the probability of classifying a positive instance correctly, given that it was predicted positive. The x-axis represents the Recall. A classifier with a PR curve closer to the upper right corner is considered as a better classifier. In contrast to ROC, a random classifier for the PR curve is a horizontal line that crosses the y-axis at $\frac{P}{P+N}$ for all threshold values, where P is the number of positives, and $P + N$ is the number of all instances in the data. For example, a random classifier for a data set with 10 positive and 90 negative examples is indicated by a horizontal line that passes through 0.1 in the y-axis, for all threshold values.

6.3 Calculated Features

In all experiments we have calculated seven node-based and one path-based metric. Specifically, the node-based features consist of *Common Neighbors, Jaccard Coefficient, Association Strength, Adamic Adar, Resource Allocation, Minimum Overlap*, and the *NMeasure*. From path-based metrics we used *Rooted pageRank*. Each measure is defined for a pair of nodes $v, w \in V_k$. The *Common Neighbors* metric calculates the common neighbors of v and w. Nodes with more common neighbors are more likely to connect in the future. *Jaccard Coefficient* is a measure that returns the similarity of two sets. If we define the neighborhoods of v and w to be those two sets, the Jaccard Coefficient calculates how similar the neighborhoods are. Values closer to 1 indicate higher similarity. *Association Strength, Minimum Overlap* and the *NMeasure* are similar to *Jaccard Coefficient*. Higher values indicate more similarity between the neighborhoods. The *Adamic Adar* metric considers in its calculation the number of neighbors of all common neighbors of v and w. The idea is that two nodes are more likely to connect in the future if their common neighbors have less number of neighbors. *Resource Allocation* is similar to *Adamic Adar* with the difference that common neighbors with larger number of neighbors are punished more heavily. Finally, *Rooted pageRank* is a restricted random walk, where the output is the likelihood that a link will form between two nodes. An extensive comparison of different baseline predictors can be found in [12].

6.4 Results

Figure 2 to Figure 8 illustrate the resulting ROC and PR curves for the seven data sets. C4.5 corresponds to the red curve, whereas HDDT to the black curve. Both types of trees exhibited similar performance in all networks. A more detailed comparison can be found in Table 3 where the AUROC metric is presented. HDDT out-performed C4.5 in `SlashDot` and `RealityCall10`, that is `RealityCall` undersampled to 1:10 ratio. Note that the results of AUROC have not been tested for statistical significance as in the case of [2]. That is, since the AUROC for both type of trees are very close to each other, the differences between them might be due to randomness.

It is clear that both classifiers perform from average to very good in the ROC space, while they perform poorly to average in the PR space, except in the case of `RealityCall10` where high precision can be observed. The poorest performance can be found in `UCIrvine60` where the AUROC is about 0.77, and the highest precision is almost 0.25 which is achieved for low values of recall. Both classifiers performed better on `Digg` and `MathOverflow` with AUROC at about 0.81 and 0.8655 respectively. Nevertheless, precision is still relatively low for the different values of recall. For `SlashDot`, the resulting AUROC is about 0.9075, with a small improvement in precision. Both classifiers performed very good in `RealityCall10` network in the case of 1:10 ratio. Finally, in the more extreme case of 1:600, the recall of the ROC curve changes slightly, whereas the precision deteriorates significantly.

An insight that we can derive by examining the plots is that, the ROC curve seems to be insensitive as the imbalance of classes becomes larger. This is clear in `UCIrvine` and `RealityCall`. As we increased the imbalance from 1:10 to 1:60, and 1:10 to 1:600 respectively, we observe minor differences in ROC, while the results of PR differ significantly.

The results of the ROC and PR curves indicate that both classifiers yield few *FN* due to high recall, but a lot of *FP* because of low precision. Thus, the probability of classifying an instance correctly as positive, given that it is positive is relatively high, but the probability of classifying an instance correctly as positive given that it was predicted as such, is relatively low. Whether or not the performance of a classifier is considered acceptable is usually domain specific. For example, when dealing with a terrorist network where the main focus is to not miss links that will form in the future, a metric such as recall would be more appropriate than precision. This is because high recall values will result to less *FN*, that is, less instances of the positive class will be classified as negatives. In other cases where *FP* are very important, the PR curve would be more informative.

	HDDT	C4.5
`UCIrvine10`	Association Strength Rooted pageRank NMeasure Resource Allocation	Association Strength Rooted pageRank Resource Allocation Minimum Overlap
`UCIrvine60`	Association Strength Rooted pageRank Minimum Overlap Resource Allocation	Association Strength Resource Allocation Jaccard Coefficient Common Neighbors
`Digg`	Association Strength Nmeasure Adamic Adar	Association Strength Miminum Overlap Rooted pageRank
`MathOverflow`	Adamic Adar Association Strength Resource Allocation	Common Neighbors Association Strength Rooted pageRank
`SlashDot`	Association Strength Common Neighbors NMeasure Adamic Adar	Association Strength Common Neighbors NMeasure Rooted pageRank
`RealityCall10`	Association Strength Adamic Adar Jaccard Coefficient	Association Strength Jaccard coefficient Adamic Adar
`RealityCall600`	Association Strength Rooted pageRank Adamic Adar	Association Strength Adamic Adar Rooted pageRank

Table 2: Most important features as indicated by the top nodes of the decision tree.

	HDDT	C4.5
`UCIrvine10`	0.7898	**0.7912**
`UCIrvine60`	0.7712	**0.7726**
`Digg`	0.8176	**0.8185**
`MathOverflow`	0.8655	0.8655
`SlashDot`	**0.9075**	0.9074
`RealityCall10`	**0.9893**	0.9888
`RealityCall600`	0.9783	**0.9814**

Table 3: AUROC for HDDT and C4.5

By examining the top nodes of the resulting trees, we get an indication of which features were the most prominent in the construction of the tree. Table 2 illustrates those features for both HDDT and C4.5. *Association Strength* was the most prominent feature in most data sets. The features are displayed with respect to importance in descending order. The less important feature is *Minimum Overlap*.

7. CONCLUSION AND FUTURE WORK

In this paper we compared HDDT and C4.5 in seven imbalanced data sets constructed from five social networks. We evaluate both trees using ROC and PR curves. Both HDDT and C4.5 exhibited similar performance, with average to good performance in ROC space, and poor to average

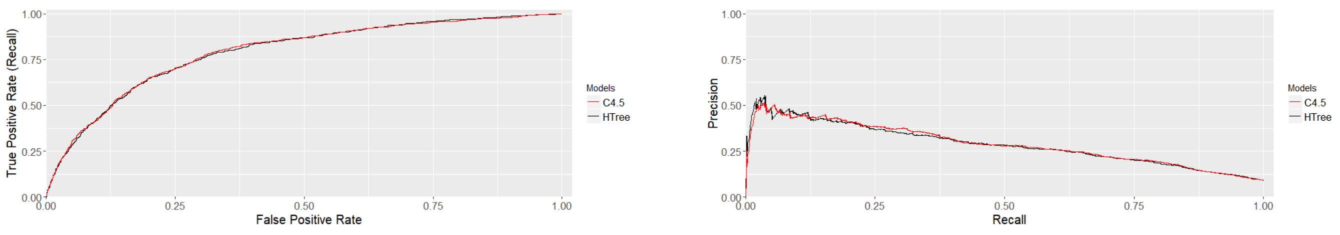

Figure 2: `UCIrvine`**, ratio 1:10**

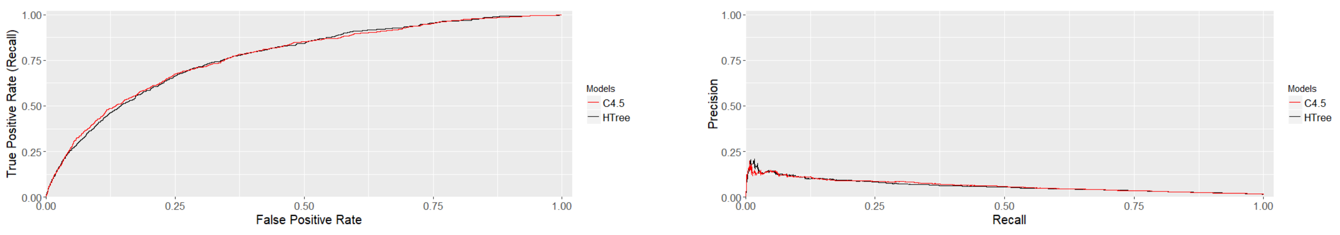

Figure 3: `UCIrvine`**, ratio 1:60**

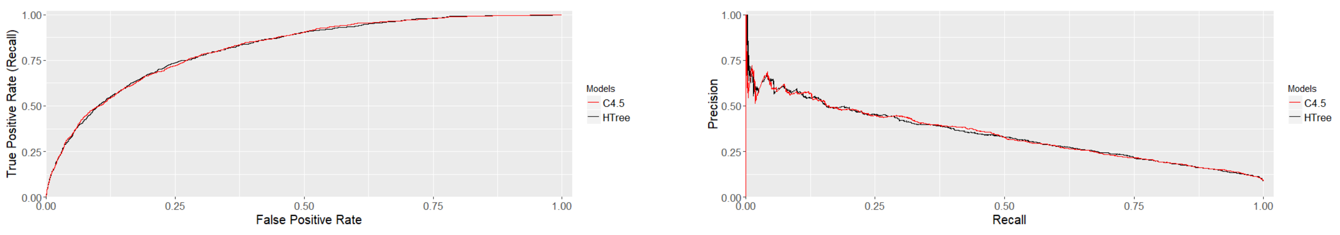

Figure 4: `Digg`**, ratio 1:10**

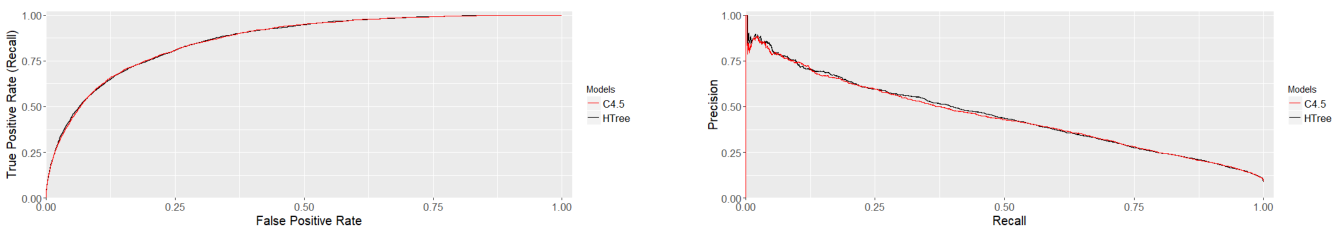

Figure 5: `MathOverflow`**, ratio 1:10**

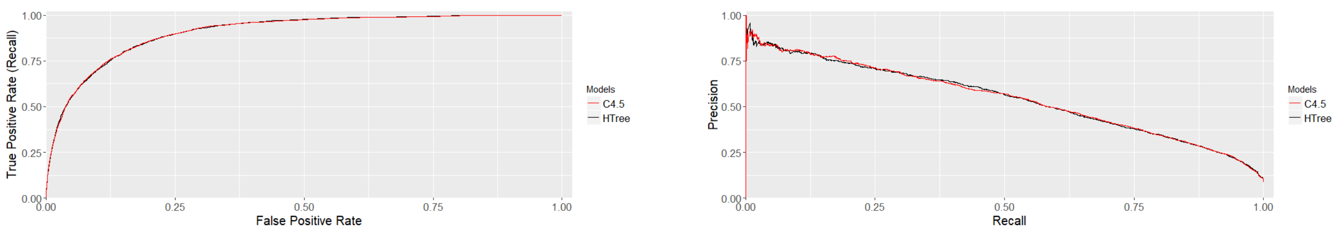

Figure 6: `SlashDot`**, ratio 1:10**

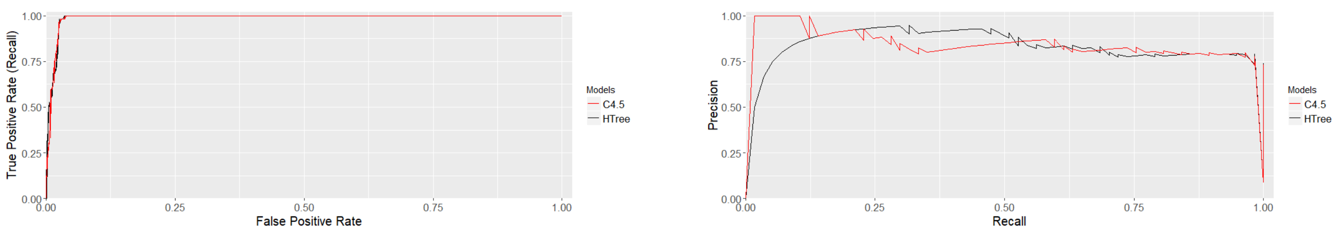

Figure 7: `RealityCall`**, ratio 1:10**

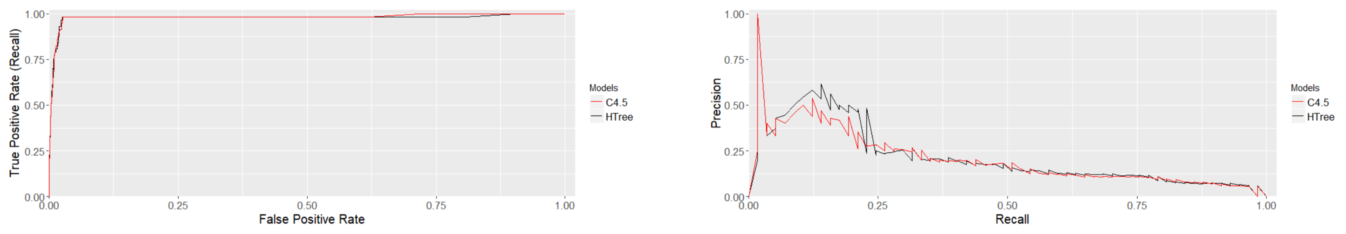

Figure 8: `RealityCall`, ratio 1:600

in the PR space. Thus, both classifiers yield high recall, but low precision. Additionally, as the class imbalance ratio increases, the precision gets lower, resulting to a larger number of *FP*. On the other hand, the recall is not influenced significantly, that is the number of *FN* does not increase. Regarding the baseline predictors that we used, by examining the top nodes of the constructed trees we found that *Associaton Strength* was the most important feature in six out of seven data sets. Our experiments were restricted to a number of chosen parameters such as the n-neighborhood size, where $n = 2$. Furthermore, we under-sample the data sets into different ratios of imbalance in order to be able to train decision trees but at the same time retain the imbalance. Further research needs to be done in order to examine the effect of choosing different predictors and the number of n-size neighborhood. Finally, different under and over-sampling techniques could be used in order to improve the predictions of classifiers.

8. REFERENCES

[1] D. A. Cieslak and N. V. Chawla. *Learning Decision Trees for Unbalanced Data*, pages 241–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[2] D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, Jan 2012.

[3] M. De Choudhury, H. Sundaram, A. John, and D. D. Seligmann. Social synchrony: Predicting mimicry of user actions in online social media. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 151–158. IEEE, 2009.

[4] H. R. De Sá and R. B. Prudêncio. Supervised link prediction in weighted networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2281–2288. IEEE, 2011.

[5] M. Dombroski, P. Fischbeck, K. Carley, et al. Estimating the shape of covert networks. In *Proceedings of the 8th International Command and Control Research and Technology Symposium*, 2003.

[6] J. Doppa, J. Yu, P. Tadepalli, and L. Getoor. Chance-constrained programs for link prediction. In *NIPS Workshop on Analyzing Networks and Learning with Graphs*, 2009.

[7] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[8] M. Fire, R. Puzis, and Y. Elovici. *Link Prediction in Highly Fractional Data Sets*, pages 283–300. Springer New York, New York, NY, 2013.

[9] V. Gómez, A. Kaltenbrunner, and V. López. Statistical analysis of the social network and discussion threads in slashdot. In *Proceedings of the 17th international conference on World Wide Web*, pages 645–654. ACM, 2008.

[10] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[11] M. A. Hasan and M. J. Zaki. *A Survey of Link Prediction in Social Networks*, pages 243–275. Springer US, Boston, MA, 2011.

[12] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[13] R. N. Lichtenwalter and N. V. Chawla. Lpmade: Link prediction made easy. *Journal of Machine Learning Research*, 12(Aug):2489–2492, 2011.

[14] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 243–252, New York, NY, USA, 2010. ACM.

[15] W.-Y. Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[16] T. Opsahl and P. Panzarasa. Clustering in weighted networks. *Social networks*, 31(2):155–163, 2009.

[17] A. Paranjape, A. R. Benson, and J. Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 601–610, New York, NY, USA, 2017. ACM.

[18] M. Pavlov and R. Ichise. Finding experts by link prediction in co-authorship networks. In *Proceedings of the 2Nd International Conference on Finding Experts on the Web with Semantics - Volume 290*, FEWS'07, pages 42–55, Aachen, Germany, Germany, 2007. CEUR-WS.org.

[19] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[20] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[21] B. Taskar, M. fai Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In S. Thrun, L. K. Saul, and P. B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 659–666. MIT Press, 2004.

[22] C. Wang, V. Satuluri, and S. Parthasarathy. Local probabilistic models for link prediction. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, pages 322–331, Washington, DC, USA, 2007. IEEE Computer Society.

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

[24] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.