Evaluating Link Prediction on Large Graphs

Dario GARCIA-GASULLA [a,1] Ulises CORTÉS [a,b] Eduard AYGUADÉ [a,b]
Jesús LABARTA [a,b]

[a] *Barcelona Supercomputing Center*
[b] *Universitat Politècnica de Catalunya - Barcelona TECH*

Abstract. Exploiting network data (*i.e.*, graphs) is a rather particular case of data mining. The size and relevance of network domains justifies research on graph mining, but also brings forth severe complications. Computational aspects like scalability and parallelism have to be reevaluated, and well as certain aspects of the data mining process. One of those are the methodologies used to evaluate graph mining methods, particularly when processing large graphs. In this paper we focus on the evaluation of a graph mining task known as Link Prediction. First we explore the available solutions in traditional data mining for that purpose, discussing which methods are most appropriate. Once those are identified, we argue about their capabilities and limitations for producing a faithful and useful evaluation. Finally, we introduce a novel modification to a traditional evaluation methodology with the goal of adapting it to the problem of Link Prediction on large graphs.

Keywords. Graph Mining, Link Prediction, Evaluation Methodology

1. Introduction

The popularization of graph-based data sets (*i.e.*, networks) in the early 21$^{\text{th}}$ century motivated research on a new family of machine learning (ML) methods. While traditional ML research has been focused on intra-entity patterns (known as the instance-attribute paradigm), the data obtained from networks generates inter-entity patterns (an instance-instance paradigm). These new methods of ML were designed to tackle network related problems such as finding the relevance of entities based on their relations (*link-based object ranking*, *e.g.*, PageRank [24] and HITS [16]), finding groups of entities strongly related (*community detection*, *e.g.*, stochastic blockmodeling [14]) or finding reoccurring association patterns (*frequent subgraph discovery*, *e.g.*, Apriori based algorithms [12]). All *graph mining* tasks having relations among entities as the cornerstone of their design.

The increased dimensionality of graph data sets often comes hand in hand with an increase in size. Together, large dimensionality and size, define the increasingly frequent family of domains known as large scale networks, which can be frequently found in social networks (*e.g.*, Twitter), biological networks (*e.g.*, brain connectome) or technological networks (*e.g.*, web graphs). Regardless of the underlying domain, computing large

[1]Corresponding Author: Dario Garcia-Gasulla, Omega building, Office 207, C/Jordi Girona, 1-3, Campus Nord UPC, Barcelona, 08034, SPAIN; E-mail: dario.garcia@bsc.es

scale networks represents a challenge in terms of efficiency, parallelism and scalability. Efficiency, because computing models and hardware architectures are not optimized for handling graph data. Parallelism, because the size of large networks makes serial approaches unfeasible. And scalability because limited computational resources constrain the applicability of exhaustive, model-based solutions. Beyond the challenges on *how* is the process implemented, the particularities of large scale networks also generate novel challenges on *what* must be done with the data from a Data Mining (DM) perspective. A prime example of that is deciding how to evaluate the performance of the mining algorithms in this novel setting.

In this paper we focus on the challenge of faithfully evaluating a graph mining task, Link Prediction (LP), when working with large scale graphs. The goal of LP is to find new or missing edges within a given graph. By using LP one can directly grow any data set represented as a graph using the same graph language (*i.e.*, adding edges among vertices by using only previous edges and vertices). As a result one could apply LP algorithms to virtually any domain that can be represented as a graph without supervision. The complexity of achieving good performance on the LP task increases with the graph size, as does the problems at faithfully evaluating performance. When a graph grows linearly in vertices, the number of possible links within the graph grows quadratically. This defines a *needle in a haystack* context where relevant or useful predictions are but a tiny fraction of all predictions. Keeping a good precision in this type of problem turns out to be very difficult, as the smallest false positive acceptance rate will amount to a huge absolute number of wrongfully predicted edges (*i.e.*, false positives). But in parallel, estimating the quality and applicability of results also becomes particularly difficult.

In §2 we explore the current solutions provided by the DM community, particularly in the context of test set construction and class imbalance. We explore the features of those methods for the particular case of LP in §3, and argue on the utility of popular approaches like ten-fold cross validation and precision-recall curves. Then in §4 we propose an addapted evaluation methodology, and argue on how to export the lessons learnt in this particular problem to other similar problems. Conclusions are presented in §5. The challenges explored in this paper refer to the LP problem on large graphs, but can be generalized to other large scale graph mining tasks.

2. Evaluation Context

LP and the rest of graph mining tasks represent a new family of DM algorithms. The particularities of these algorithms originate from the special nature of networks. Particularities that include data dimensionality, variable dependency, and often log-scale distribution of information. Even with these differences, one can find analogies between graph mining problems and general DM problems. From a traditional DM perspective, LP can be reduced to a binary classification problem between two classes: the positive class of edges that do or should exist, and the negative class of edges that do not and should not exist. Given a directed graph $G = (N, E)$, and all the possible edges in the graph (of size $|N * (N-1)|$), the problem of LP would be that of distinguishing between those edges that exist, $e \in E$, and those that do not, $e \notin E$. The analogy between LP and binary classification is accurate in most cases, as the target of LP is often to identify the positive class. Which in terms of graph mining is equivalent to finding and propos-

ing missing links. For the remaining of this paper we will assume this mainstream case. In the bibliography there are a many methodologies available for the evaluation of this type of problem. These methodologies are typically discriminated based on the problem characteristics, which in the case of the LP classification problem are dominated by *class imbalance*.

2.1. Test Sets

To evaluate a binary classifier empirically we require a test set. Given a graph, LP algorithms can propose a number of edges to be added to it, however, to validate the quality of those proposals, we need a set of edges known to be correct and missing from the graph. In evaluation, each predicted edge found in the test set is considered as a correct prediction, while each predicted edge not found is considered as a mistake. From these results one can then obtain performance indicators like *precision* and *recall*.

The main problem with tests sets is how to obtain them. In the case of LP, the best test set one can use is that which represents a natural extension of the graph being computed. This is feasible on temporally grounded domains. For example, for a graph composed from Wikipedia articles and the hyperlinks among them from 2012, we can obtain a natural test set by considering the links added to Wikipedia between 2012 and 2013 [10]. Unfortunately, the domains and graphs having such incremental nature are rare. Instead, in most cases one must settle for the more drastic approach of randomly removing a number of edges from the graph in order to use them as test set. A frequent concern when one must split a set of data to produce a test set is representativity. Typically, a random split cannot guarantee a prototypical distribution. The most frequent solution for avoiding bias is ten fold cross validation (10-fold CV). Within the LP problem, splitting data to build a test set will be often necessary [25]. However, as we will see in §3.2 performing 10-fold CV is redundant.

2.2. Class Imbalance

To reduce the LP problem to a binary classification problem one needs to categorize edges in two classes. However, the number of edges that exist is typically but a small fraction of the edges that do not exist. In fact, in real world networks the average vertex degree is rarely over fifty [19,21,1], a feature that is consistent even as graphs grow to billions of vertices [27]. From a classification problem perspective, this implies a class imbalance, as the negative class is very large in comparison to the positive class. As it is well known, class imbalance can be a severely complicating factor in classification problems [3,20,26,28].

The degree of class imbalance found on large graphs when performing LP is hard to overestimate, and it only gets worse as graphs grow. In fact, examples with a similar degree of class imbalance are rarely found in the bibliography. In LP, by adding vertices to a graph one increases the size of the possitive class linearly ($N * k$), since the average vertex degree remains stable. Nevertheless, in parallel, the negative class grows quadratically ($N * (N - 1)$). Consequently, class imbalance in LP grows linearly with the number of vertices. To illustrate on the degree of class imbalance, Table 1 shows the topological properties of some real world graphs obtained from WordNet [9], the Cyc project [8], the movie-related IMDb knowledge base [10], and several web graphs from the Notre-

Dame University [2], Stanford/Berkley universities [15], a Google challenge [10], and the Hudong, Baidu [17] and Wikipedia encyclopedias [18]. Notice how, in the *best* case scenario, the class ratio is of 1 positive instance for every 11,382 negative instances.

The impact of class imbalance on classifiers was explored in [13], and authors concluded that this impact was largely reduced when all classes were of reasonable size. *A priori* this should be good news for LP on large graphs, as its classes seem to be of *reasonable* size; the positive class of all graphs shown in Table 1 is over 10,000 entities. Unfortunately, this assumption does not apply to the LP problem [19], and the reason for this is twofold. On one hand the imbalance found in LP on large graphs is several orders of magnitude larger than any imbalance tested in [13]. Thus its impact may remain significant. On the other hand, LP is not a typical classification problem, and given the small amount of information provided by each edge (*e.g.*, positive instances have no attributes), a class composed 30,000 elements could still be considered to be small. In reality, class imbalances of 1:10,000 or larger translate as a strong tendency towards false positive classification mistakes, as incorrectly accepting negative instances becomes almost inevitable. The main challenge of LP is therefore precision, a notion that should be taken into account by the evaluating methodologies.

2.3. Evaluation under Class Imbalance

A frequent approach of supervised or semi-supervised learning methods to overcome class imbalance is to equilibrate the training set through over-sampling, under-sampling or feature selection [3,20,26,28]. Unsupervised LP algorithms cannot benefit from these solutions as adding or removing edges from the data set would equal to perform arbitrary classification, and there are no features to be removed beyond the existence of edges among vertices.

The most frequently used methods for classifier evaluation are based on accuracy. However, these methods are biased towards the classification of instances within the large class, making them inappropriate for imbalanced data sets [3,20,26,11]. Using them for LP would be almost analogous to measuring the capability of algorithms at predicting which edges should not be added to the graph, which is not the goal of LP. For data sets with large class imbalance, the most frequently used methodology is the Receiver Operating Characteristic (ROC) curve and the derived Area Under the Curve (AUC) measure [6]. The ROC curve sets the True Positive Rate (TPR) against the False Positive Rate (FPR), making this metric unbiased towards entities of any class regardless of their size.

Data source	Number of vertices	Average edges per vertex	positive:negative class ratio
WordNet	89,178	15.66	1:11,382
Cyc	116,835	5.9	1:39,496
webND	325,729	9.18	1:70,867
webSB	685,230	22.18	1:61,775
webGL	875,713	11.64	1:150,217
hudong	1,984,484	14.98	1:264,848
baidu	2,141,300	16.72	1:257,667
IMDb	2,930,634	5.12	1:1,140,835
DBpedia	17,170,894	19.44	1:2,151,672

Table 1. Sample of average number of edges per vertex and class imbalance on real graphs

The AUC measures the area below the curve in order to compare the overall predictive performance of two different curves.

ROC curves are unbiased in imbalanced contexts, but their consideration of miss-classifications can result in mistakenly optimistic interpretations [5,25]. When the negative class is very large, showing mistakes as relative to the negative class size (*i.e.*, FPR) can hide their actual magnitude, and make it complicated to assess the overall performance quality. From a practical perspective, most of the ROC curve is irrelevant when dealing with large class imbalance, as it represents completely unacceptable precisions. For example, one may consider that a classifier achieving a TPR of 0.95 (finding 95% of all positive edges) and a FPR of 0.01 (incorrectly accepting 1% of all negative edges) in the ROC curve demonstrates an excellent performance. However, for a data set with a positive:negative rate of 1:100 those results imply that the classifier accepts more negative edges than positive edges (*i.e.*, it has a precision smaller than 0.5). For domains with a 1:11,000 or worse ratio, like the ones shown in Table 1, the limitations of the ROC curve become even more striking. In those even a FPR of 0.0001 implies a very poor precision/performance regardless of the TPR achieved.

Precision-recall (PR) curves are an alternative to ROC curves. A PR curve is resistant to class imbalances as it focuses only on the performance achieved for the positive class (typically the small one), and does not show the number of correct classifications for the negative class. ROC and PR curves are strongly related; a curve dominates another (it is above it) in the ROC space if and only if it also dominates it in PR space [5]. The main difference between ROC and PR curves is on how errors are represented. While ROC curves show miss-classifications as relative to the total number of negative cases, PR curves show miss-classifications as relative to the total number of predictions done. PR curves plot precision (y axis) against recall (x axis), thus focusing on high precision predictions. As an illustration on the impact of these differences, consider how the two curves represent a random classifier, which always performs poorly in an imbalanced data set. The ROC curve always represents the random classifier as a straight line between points $(0, 0)$ and $(1, 1)$, regardless of class imbalance, with all better than random classifiers rep resented as lines above that diagonal. PR curves on the other hand represent random classifiers in imbalanced data sets a flat line on the x axis, as their precision in imbalanced settings is always close to zero.

3. Evaluating Link Prediction

Current solutions for performance evaluation, like the ones shown in §2, have severe limitations when applied to large graph mining problems. Issues like test set representativity, or the evaluation under class imbalance, reach a new degree of relevance when considering problems like LP on large networks. In this section we discuss these problems in depth and propose solutions fitting our LP problem.

3.1. Representativity of Test Sets

The use of a test sets to evaluate LP implies the assumption that the test set (the prediction of which is evaluated by the curves) faithfully represents the *correct* edges missing from the graph. Or in other words, that all edges not found in neither the graph nor in the

test set, are wrong. In certain cases, where the graph topology is stable, this may be an accurate assessment. For example, a graph obtained from WordNet data (as shown in [9]) can be considered as almost perfect, as WordNet relations have been identified, discussed and implemented by linguists for years. In other cases though test sets are an imperfect measure of the right edges missing from the graph. Consider for example a graph obtained from Wikipedia articles and hyperlinks, in which the pagelinks among Wikipedia articles from 2012 are used as training and the new pagelinks added on 2013 are used as test. This graph is clearly incomplete, as new links are being added every day. The Wikipedia grows continuously and the fact that a link is not implemented so far does not mean it is wrong. As a result, one must take into account that some of the edges predicted, not found in the test set and labeled as mistakes, will in fact be correct predictions corresponding to edges not yet added to the graph.

Using a test set which does not fully represent the target class implies an underestimation of performance, as the predictions being made outside of the test set will always (and not always correctly) be considered as mistakes. Nevertheless, since this limitation applies to all the methods being evaluated (assuming all methods are evaluated using the same test set), it can be argued that the resultant performance indicators remain valid for comparative purposes. That is, we can still find out which LP algorithm works better. The unavoidable shortcoming of representativity comes when evaluating the precision of a score in the context of applicability. That is, we cannot be sure of how well performs the best LP algorithm. The only way to obtain a faithful, non-comparative evaluation of performance of a single LP algorithm would be a hand-made validation. One could achieve an approximate solution by performing a sampling process of all edges predicted, manually evaluating the sampled edges as correct or incorrect predictions, and then extrapolating the performance obtained on the sample to the rest of the graph. There are several aspects to keep in mind with this solution. First of all, the sampling needs to be large for the extrapolation to be faithful, which equals to many hours of manual labeling. And second, the sampling would have to be done at several thresholds so that extrapolations are representative of the whole curve. Sampling may therefore be the only accurate evaluation methodology for estimating predictive performance of a given score on a specific domain, at the price of a huge amount of manual labeling hours.

3.2. 10-fold CV

10-fold CV is a commonly used technique for reducing variance in test set construction and improving representativeness. Although 10-fold CV is almost universally expected when using test sets that are a random portion of a complete data set, we argue that it is not needed when performing large graph mining. The main reason behind that argument being the large size of these domains, which naturally avoid variance. To asses the utility of 10-fold CV we test a webgraph obtained from a Google challenge, composed by 875,713 vertices and 5,105,039 edges. This particular graph could be considered to be medium sized, as it is easy to find much larger ones (see Table 1). The conclusions obtained for this graph could be extended, even with more reliability, to larger graphs. A random 10% test set of the Google challenge webgraph is composed by 510,503 edges. We obtain ten different random test sets from this graph, and use each one of them to evaluate seven different LP algorithms. As a result we obtain ten PR curves (as these are preferred to ROC curves, see §2.3), for seven different LP algorithms. In Figure 1 we

Algorithm	Minimum AUC	Maximum AUC	Mean	Standard Deviation
#1	0.0892558	0.0899991	0.08971357	0.0002287783
#2	0.10017	0.10083	0.1005145	0.0001902058
#3	0.0618143	0.0625483	0.06225763	0.0002040158
#4	0.128201	0.128857	0.1285072	0.0001879318
#5	0.124934	0.125385	0.1251525	0.0001210622
#6	0.419577	0.421239	0.4204078	0.0004921798
#7	0.491902	0.4935	0.4925985	0.0005283041

Table 2. Using the Google challenge webgraph, AUC obtained by seven different algorithms on ten different random test sets.

shown the ten curves belonging to one of those algorithms, to show the minimal variance found among curves. The ten curves are virtually identical, which implies that variance among random splits is irrelevant. To empirically validate this assertion, in Table 2 we show the AUC of the seventy curves obtained, ten for each algorithm. Results show that the variance of 10 executions using 10 randomly selected tests sets is very low. In fact, the standard deviation represents a 0.32% of the mean value in the worst case (algorithm #3). Such a low variance is the result of having a large test set, which, given the law of large numbers, will tend towards a stable sample. In this context it seems clear that performing 10-fold CV is not necessary, as a single run is a representative and accurate sample of the performance.

Regardless of these results, performing 10-fold CV is not a wrong or misguiding strategy. Our argument here is that 10-fold CV is not required in order to consider some results representative. This fact is particularly relevant due to the computational cost of computing large scale graphs. Building test sets and running graph mining algorithms on them is typically expensive in computational terms. Hence, the physical resources and time spent doing ten equivalent executions could be use more efficiently elsewhere.

3.3. Precision-Recall Curves in Link Prediction

Most research on LP use ROC curves[22,7,4,21] or PR curves [23,1] for evaluation, but for the reasons discussed in §2.3 we find PR curves to be more appropriate. PR curve shows the performance of a classifier at various thresholds: at the left part of the curve are the high-certainty predictions where precision is higher, while at the right part of the curve are low-certainty predictions where recall grows at the expense of a lower precision. Through the PR curve one can see which classifier performs better at each threshold. The derived AUC metric of the PR curve on the other hand determines which classifier performs better overall, when all thresholds are considered at the same time with the same importance. Due to this last point, we find the PR-AUC score to be sub-optimal for evaluating the applicability of results. Given the imbalance of the graphs used (see Table 1), a large part of the PR curve represents very low precisions. As recall grows precision can quickly reach levels unacceptable from a practical point of view. At this point one must consider which results are worth taking into account when evaluating performance. If we intend to achieve an applicable methodology we should focus on its performance where it matters, when a *reasonable* number of mistakes are being done. At extremely low precisions (*e.g.*, 0.01%) results are likely to be useless, and therefore should not be taken into account (certainly not with equal weight) into the evaluation. For this reason in §4 we propose a AUC measure focusing on applicability.

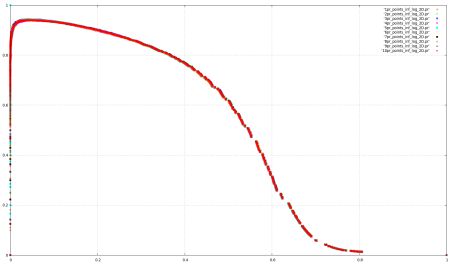

Figure 1. Precision-Recall curve of a LP algorithm using 10 different random splits.

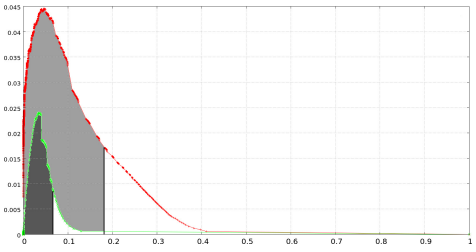

Figure 2. PR curve of a LP score on two different graphs. Grey area shows the CAUC.

4. Constrained AUC

One of the main goals of LP is to produce high certainty and high utility predictions. In this context, in order to be successful LP does not need to classify most edges correctly. LP needs only to correctly classify a significant set of edges with high certainty to become useful. Instead of building two clearly delimited classes of edges, LP ought to focus on building one high-certainty class of edges for which its existence is well founded. We consider the necessity of evaluating performance in the context of applicability, so that we can determine which LP score produces the most useful results. The classic AUC measure equally considers the performance of a classifier at any threshold, therefore evaluating which classifier performs the best overall. However, we consider that the performance of LP scores at low thresholds, where millions of mistakes are done, is irrelevant for assessment purposes.

We propose to evaluate LP scores based only on the predictions produced while keeping an acceptable ratio of mistakes. In order to be fair with the particularities of each domain, we define this *acceptable* ratio of mistakes based on the number of edges originally found in the graph. The more edges there are in a graph, the more predictions we expect to obtain, therefore extending the limit of mistakes considered as *acceptable*; we assume that one will rarely want to predict more new edges than edges already found in the graph. Formally, the proposed Constrained AUC score (CAUC) is obtained by calculating the AUC of the PR sub-curve where the number of non-existing edges mistakenly accepted by the score is equal or lower than the total number of edges in the graph.

The CAUC considers only the part of the PR curve where the score is incorrectly accepting less edges than the number of edges in the graph, dismissing the rest of the area under the curve. CAUC therefore calculates a portion of the AUC starting from the left of the curve, ending when too many mistakes are done. Nevertheless, if a LP score is precise enough the CAUC can be equal to the AUC. Notice that, unlike the AUC, the CAUC can be 0, if the top E edges evaluated by a score in a graph with E edges are incorrect predictions.

As said before, the goal of this performance measure is to focus on the relevant parts of the PR curve. By accepting only mistakes up to the number of edges in the graph one puts a limit to what one considers are potentially useful predictions. Thus, scores which perform better at low precisions are penalized by this score, whereas they are not by the AUC. Furthermore, by making the threshold dependent on graph properties (*i.e.*, on the number of edges in it) the CAUC score adapts to domain specific properties such as sparsity and graph size. This is a interesting novel feature not found in the AUC measure,

as a contextual evaluation allows the cross-domain comparison of results. As an example consider Figure 2, where the PR curves of a LP score are shown for two different graphs. The vertical cut on each curve represents the location of the CAUC threshold for each particular data set and score, limiting the CAUC to the area at the left of the threshold (colored in grey), whereas the AUC considers the whole curve.

A relevant feature of the CAUC measure is that it adapts to the graph under evaluation. CAUC defines the threshold stating which predictions are relevant and which are not according to each graph size. Simply put, CAUC does not take into account predictions made after more mistakes are done than actual edges in the graph. As a result, more predictions will be demanded for graphs with more edges. AUC on the other hand evaluates the predictions done on a graph with N vertices and 1000 edges and the predictions done on a graph with N vertices and 100,000 edges under the same conditions, as if these two problems were equally difficult. A clearly unrealistic assumption that may lead to the underestimation or overestimation of results.

5. Conclusions

In LP, as in many other graph mining tasks, one must face certain properties unprecedented in DM. Data set size and class imbalance are the most important features in terms of evaluation methodologies. Data set size for example complicates the use of exhaustive testing policies such as 10-fold CV, due to an increased computational cost. However, the same data set size fixes this problem, as 10-fold CV *may* not be needed in most cases due to the little variance there is in a random sample of a large graph. As for class imbalance it forces to use evaluation unbiased by imbalance methods if one wish to obtain meaningful results. However, class imbalance, in some cases, is so large that even unbiased methods like ROC curves can produce miss-informative results.

Class imbalance impacts classification performance, typically by reducing precision. This condition has constrained the broad applicability of LP so far. However, in our opinion, most performance measures like the AUC-PR or AUC-ROC are not a reliable measure when evaluating LP algorithms. Since the test set is so large, one can focus on high confident predictions at the cost of a smaller recall, and still obtain thousands of reliable predictions with *good* precision. Which is the goal of LP in most applied cases. Following this notion we proposed a constrained version of the AUC-PR measure (CAUC), focusing on high confidence predictions. This proposal follows the common sense idea that through LP one does not intend to generate more links than the ones previously existing. The CAUC adapts to the size and sparsity of every graph, serving thus as a comparative measure.

References

[1] Aggarwal, C. C., Xie, Y., and Yu, P. S. (2013). A framework for dynamic link prediction in heterogeneous networks. Statistical Analysis and Data Mining.

[2] Albert, R., Jeong, H., and Barabási, A.-L. (1999). Internet: Diameter of the world-wide web. Nature, 401(6749):130-131. 51

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16:321-357.

[4] Clauset, A., Moore, C., and Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks. Nature, 453(7191):98-101.

[5] Davis, J. and Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In Proceedings of the 23^{rd} Int Conf on Machine learning, pages 233-240. ACM.

[6] Fawcett, T. (2004). ROC graphs: Notes and practical considerations for researchers.

[7] Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L., and Elovici, Y. (2011). Link prediction in social networks using computationally efficient topological features. In 2011 IEEE 3^{rd} international conference on social computing, pages 73-80.

[8] Garcia-Gasulla, D. and Cortés, U., Hierarchical inference on semantic graphs applied to CyC, Proceedings of the 16th International Conference of the Catalan Association of Artificial Intelligence, 2013.

[9] Garcia-Gasulla, D. and Cortés, U., Link Prediction in Very Large Directed Graphs: Exploiting Hierarchical Properties in Parallel, 3^{rd} Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data - $11t^{th}$ Extended Semantic Web Conference, 2014.

[10] Garcia-Gasulla D (2015). Link Prediction in Large Directed Graphs. PhD Dissertation, Universitat Politècnica de Catalunya - Barcelona TECH.

[11] He, H. and Garcia, E. (2009). Learning from Imbalanced Data. Knowledge and Data Engineering, IEEE Transactions on, 21(9):1263-1284.

[12] Inokuchi, A., Washio, T., and Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. In Principles of Data Mining and Knowledge Discovery, 13-23. Springer.

[13] Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent data analysis, 6(5):429-449.

[14] Karrer, B. and Newman, M. E. (2011). Stochastic blockmodels and community structure in networks. Physical Review E, 83(1):016107.

[15] Khalil, A. and Liu, Y. (2004). Experiments with PageRank computation. Indiana University, Department Computer Science. URL: http://www.cs.indiana.edu/akhalil/Papers/pageRank.pdf.

[16] Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM), 46(5):604-632.

[17] Kunegis, J. (2013). KONECT: the Koblenz network collection. In Proceedings of the 22^{nd} international conference on World Wide Web companion, pages 1343-1350. International World Wide Web Conferences Steering Committee.

[18] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2014). DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web Journal.

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

[20] Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2009). Exploratory undersampling for classimbalance learning. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 39(2):539-550.

[21] Lü, L., Jin, C.-H., and Zhou, T. (2009). Similarity index based on local paths for link prediction of complex networks. Physical Review E, 80(4):046122.

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

[23] Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 809-816.

[24] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: bringing order to the web.

[25] Yang, Y., Lichtenwalter, R. N., and Chawla, N. V. (2014). Evaluating link prediction methods. Knowledge and Information Systems, pages 1-32.

[26] Wasikowski, M. and Chen, X. W. (2010). Combating the small sample class imbalance problem using feature selection. Knowledge and Data Engineering, IEEE Transactions on, 22(10):1388-1400.

[27] Watanabe, M. and Suzumura, T. (2013). How social network is evolving? A preliminary study on billion-scale twitter network. In Proceedings of the 22^{nd} Int Conf on World Wide Web companion, pages 531-534. Int WWW Conferences Steering Committee.

[28] Weiss, G. M. (2004). Mining with rarity: a unifying framework. ACM SIGKDD Explorations Newsletter, 6(1):7-19.