Link Prediction in Very Large Directed Graphs: Exploiting Hierarchical Properties in Parallel

Dario Garcia-Gasulla[1] and Ulises Cortés[1]

Knowledge Engineering and Machine Learning Group (KEMLg), Universitat Politècnica de Catalunya (UPC) - Barcelona Tech, Jordi Girona 1-3, UPC, Campus Nord, Office Omega-207, Barcelona, SPAIN, 08034. Tel.:(+34)934134011
dariog@lsi.upc.edu, ia@lsi.upc.edu

Abstract. Link prediction is a link mining task that tries to find new edges within a given graph. Among the targets of link prediction there is large directed graphs, which are frequent structures nowadays. The typical sparsity of large graphs demands of high precision predictions in order to obtain usable results. However, the size of those graphs only permits the execution of scalable algorithms. As a trade-off between those two problems we recently proposed a link prediction algorithm for directed graphs that exploits hierarchical properties. The algorithm can be classified as a local score, which entails scalability. Unlike the rest of local scores, our proposal assumes the existence of an underlying model for the data which allows it to produce predictions with a higher precision. We test the validity of its hierarchical assumptions on two clearly hierarchical data sets, one of them based on RDF. Then we test it on a non-hierarchical data set based on Wikipedia to demonstrate its broad applicability. Given the computational complexity of link prediction in very large graphs we also introduce some general recommendations useful to make of link prediction an efficiently parallelized problem.

1 Introduction

Graphs have become a frequently used structure for Knowledge Representation (KR). The task of extracting knowledge from graphs however has only become popular recently, motivated by the availability of very large, multi-dimensional data sets. One of the most active fields in this line of research is link mining [Getoor and Diehl, 2005], which comprises all tasks *building predictive or descriptive models of linked data*. Link mining includes problems such as link-based object ranking (*e.g.*, PageRank [Page et al., 1999], HITS [Kleinberg, 1999]), group detection (*e.g.*, through stochastic blockmodeling [Karrer and Newman, 2011]) and frequent subgraph discovery (*e.g.*, Apriori based algorithms [Inokuchi et al., 2000]). Among those, Link Prediction (LP) is as a particularly interesting one, as it directly enriches existing graphs by adding new edges.

Similarity-based algorithms are the most scalable approach to LP [Lü and Zhou, 2011]. These algorithms score each pair of nodes independently to estimate the likelihood of their link. The rest of approaches to LP require a model of the whole graph, a model which becomes unfeasible to compute as the graph grows. Among similarity-based algorithms the ones that scale the best are those using information local to the pair of nodes (*i.e.*, its direct neighbours) in order to calculate each link score. For graphs with hundreds of thousands, or even millions of nodes, these algorithms are the only ones that scale well enough as to be a feasible solution nowadays. In this context we recently proposed a local similarity-based algorithm to LP [Garcia-Gasulla and Cortés, 2014]. Our algorithm differs from the rest of its kind in that it assumes the existence of a local model in the graph (a hierarchy) so that it does not need to compute it. To predict links based on that model the algorithm estimates the likelihood with which a node is *below* another one in the graph given their local neighbours. Although our algorithm requires the graph to be directed while most LP proposals work for undirected graphs, nowadays there are as many available directed graphs as there are undirected.

Making of LP in large, sparse graphs a profitable problem is complicated, as the huge class imbalance strongly penalizes any lack of precision. At the same time, the cost of computing very large graphs requires that applied algorithms scale well, which can affect their precision. Between these two problems, our goal is to obtain a scalable methodology that can be easily applied to a variety of relevant and large data sets, and that is precise enough to be broadly applicable. To do so, first we compare our algorithm with the current best algorithm of its kind on two large graphs with clear hierarchical semantics. After that we test our algorithm on a very large graph (obtained from the Wikipedia) which does not implement a hierarchical structure. The rest of the paper is organized as follows, in §2 we review the related work in LP. In §3 we discuss and introduce our proposal. In §4 we present a performance analysis on three different data sets. Then, in §5 we outline our parallelization method and discuss some efficiency issues of LP. Finally, in §6 we present our conclusions.

2 Related Work

There are three main approaches to LP [Lü and Zhou, 2011]: similarity-based algorithms, maximum likelihood algorithms and probabilistic models. The last two need to build and compute a model for the whole graph, which becomes unfeasible once the graph reaches a certain size (*e.g.*, millions of nodes). Probabilistic models are typically based on Markov and Bayesian Networks, while maximum likelihood algorithm assume and compute a certain model for the whole graph (often hierarchical or community based). An interesting example of maximum likelihood algorithm is the Hierarchical Random Graph [Clauset et al., 2008].

The third family of LP methods are similarity-based algorithms. These algorithms compute a measure of distance or score between each possible pair of nodes within the graph. This score is then used to determine the likelihood of each possible edge within the graph. Even for this simpler type of algorithms there are solutions which cannot be scaled to graphs with millions of nodes. If the information used to obtain each score is global, *i.e.*, it is derived from the complete graph topology, the cost of fully traversing the graph quickly becomes prohibitive [Lü and Zhou, 2011]. On the other hand, if the information used is local, *i.e.*, it is derived from the direct neighbours of the pair of nodes, the reduced cost allows one to compute extremely large graphs.

Quasi-local indices are a compromise between global an local scores. These indices execute a variable number of steps, typically increasing the depth of the graph being explored. As the number of steps grows, the algorithm becomes more expensive (often exponentially [Lü and Zhou, 2011, Liu and Lü, 2010]). The most popular quasi-local indices are based on the random walk model [Liu and Lü, 2010] and on the number of paths between the pair of nodes [Lü et al., 2009]. In essence quasi-local indices originate from local scores: when the minimum number of steps is set quasi-local indices are equivalent to a local score. For example, local path index reduces to the local score common neighbours [Zhou et al., 2009], while superposed random walk and local random walk reduce to the local score resource allocation [Liu and Lü, 2010]. Beyond the cost of computing a larger part of the graph, the sampling process required by quasi-local indices to determine the optimum number of steps to be performed also adds a significant overhead. In that regard, no test has been performed so far to evaluate the performance and cost of quasi-local indices on graphs with millions of nodes.

Finally, let us mention a completely different family of algorithms which have come close to the field of LP are tensor factorization algorithms. These algebraic methods were first used for link-based object ranking of entities extracted from RDF triplets [Franz et al., 2009], and have also been used to obtain a score for non existing triplets in a given KB [Drumond et al., 2012].

2.1 Local similarity-based algorithms

Similarity-based algorithms were first evaluated on five different scientific co-authorship graphs in the field of physics [Liben-Nowell and Kleinberg, 2007]. Three scores consistently achieved the best results in all data sets: local algorithms Adamic/Adar (AA) and Common Neighbours (CN), and global algorithm Katz. In [Murata and Moriyasu, 2008] similar results were obtained, with AA and CN achieving the best results among local algorithms. In [Zhou et al., 2009] a new local algorithm called Resource Allocation (RA) was proposed and compared with other local similarity-based algorithms. Testing on six different datasets showed once again that AA and CN provide the best results among local algorithms, but it also showed that RA could improve them.

In [Garcia-Gasulla and Cortés, 2014] we evaluate AA, CN and RA on two of the data sets used here (Cyc and Wordnet), with RA clearly outperforming the others. We therefore decided to use RA as a baseline for our algorithm in §4. The RA algorithm is based on the resource allocation process of networks. In its simpler implementation each node transmits a single resource unit, having this resource evenly distributed among its neighbours. In this case, the similarity between nodes x and y becomes the amount of resource obtained by y from x

($\Gamma(x)$ represents the set of nodes connected with x)

$$s^{RA}_{x,y} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{|\Gamma(z)|}$$

Most LP algorithms are currently designed for undirected graphs, disregarding all information related with directionality. In practice that means scores like RA cannot separately characterize the pair of edges $x \to y$ and $y \to x$ (*i.e.*, $s_{x,y} = s_{y,x}$). To mitigate this handicap so that we can test RA against our own algorithm under similar conditions, we adapt RA to directed graphs following our own hierarchical approach. We make RA consider only those edges going **from** specific nodes **towards** generic nodes. We define how specific a node is as the number of nodes below it in the hierarchy (*i.e.*, the number of nodes each node can be reached from). In previous work [Garcia-Gasulla and Cortés, 2014] we saw how this modification consistently improves the performance of undirected similarity-based algorithms in hierarchical graphs (CN, AA and RA all improved their results this way). Formally, the Hierarchical Resource Allocation score (HRA) is defined as

Definition 1.

$$s^{HRA}_{x \to y} = \begin{cases} s^{RA}_{x,y} & \textit{if } x \textit{ is more or equally specific than } y \\ 0 & \textit{if } x \textit{ is more abstract than } y \end{cases}$$

3 Hierarchical Link Prediction

Our proposed LP score is based on the concept of hierarchy, one of the most general structures, if not *the* most general structure, used for knowledge organization. The core semantics of hierarchies (*e.g.*, generalization and specialization) are found in domains as diverse as protein structure or terrorist organizations, and constitute backbone core of most KR schemes. Within Knowledge Bases (KB) hierarchical properties are found in a variety of ways: through the linguistic relation *hyponym/hypernym*, in ontologies through the *is_a* relation, in RDFS through relations such as `rdf:type` and `rdfs:subClassOf`, *etc.*. Regardless of the method used to represent hierarchies, the universal semantics of these structures makes them essential for representing concepts more complex than a hierarchy itself.

For evaluating the importance of hierarchical properties in knowledge definition, in [Garcia-Gasulla and Cortés, 2014] we presented INF, a hierarchy-based LP score (introduced next in §3.1). We tested its predictive capability on two semi-formal domains which contained explicit representations of a hierarchy: Wordnet and Cyc. We will introduce these results in §4.1 and §4.2. Next we intend to evaluate the importance of hierarchical properties in domains with no explicit hierarchical structure. Our hypothesis is that most directed graphs contain a sense of hierarchy in them which can be exploited as defined by edge directionality: while undirected edges represent relations between nodes, directed edges represent asymmetric relations, the source of hierarchical structure.

3.1 INF Score

According to our interpretation of a hierarchy, generalization and specialization are the two main semantic properties available for each node. We impose only two restrictions on what generalization and specialization actually represent *w.r.t.* the relation between elements. First, **an element is partly defined by the elements it generalizes, as it somehow reduces their semantics.** And second, **an element is partly defined by the elements it specializes, as it somehow aggregates their semantics.**

Given an element x, we name the elements that generalize x the *ancestors* of x ($A(x)$), and the elements that specialize x the *descendants* of x ($D(x)$). Mapping these concepts into directed graphs is straightforward, as $A(x)$ represents the set of nodes destination of an edge originated in x, and $D(x)$ the set of nodes origin of an edge which ends in x. Considering $\rightarrow$ as the directed edge of a graph $G = (V, E)$ we formalize the ancestors and descendants sets

Definition 2.

$$\forall x, y \in V : x \in D(y) \leftrightarrow x \rightarrow y \in E$$

$$\forall x, y \in V : x \in A(y) \leftrightarrow y \rightarrow x \in E$$

At this point we define the hierarchical scores we propose to predict links. We start with one suggested by the following deductive reasoning and supported by the generalizations of a node: if most of my parents are mortal, I will probably be mortal too. Or in other words, if most of my ancestors share an edge, I should probably share it as well. We call this the the *deductive* score (DED for short)

Definition 3.

$$s^{DED}_{x\rightarrow y} = \frac{|A(x) \cap D(y)|}{|A(x)|}$$

The second main hierarchical score we use is suggested by the following inductive reasoning and supported by the specializations of a node: if most of my children are mortal, I will probably be mortal too. In other words, if most my descendants share an edge, I should probably share it as well. We call this the the *inductive* score (IND for short)

Definition 4.

$$s^{IND}_{x\rightarrow y} = \frac{|D(x) \cup D(y)|}{|D(x)|}$$

We combine the DED and IND scores within a single score by adding them. This produces a hierarchical affinity measure for each possible pair of nodes based on the combined evidence of their generalizations and specializations. This algorithm we called the *inference* score (INF for short).

Definition 5.

$$s^{INF}_{x\rightarrow y} = s^{DED}_{x\rightarrow y} + s^{IND}_{x\rightarrow y}$$

In this implementation of INF *i.e.*, only those elements directly connected with x compose $A(x)$ or $D(x)$. Consequently, according to the definitions given here DED, IND and INF are local similarity-based scores. However, like the quasi-local indices discussed in §2, our proposed score can be extended to a quasi-local index by executing a variable number steps. These steps would allow the algorithm to consider further nodes, simply by extending Definition 2 to include within $A(x)$ and $D(x)$ nodes reachable at a larger distance.

4 Evaluation

LP algorithms are often evaluated using the Receiver Operating Characteristic (ROC) curve [Murata and Moriyasu, 2008, Lü and Zhou, 2011]. This metric compares the False Positive Rate (FPR, in the x axis) with the True Positive Rate (TPR, in the y axis) at various thresholds. In the ROC curve the straight line between points [0,0] and [1,1] represents the random predictor; the function defined by points [0,0], [0,1] and [1,1] represents the perfect classifier. We randomly split our first pair of graphs to build the ROC curve. 90% of edges will be used as input for the algorithms and the remaining 10% will be used to evaluate them. To evaluate the third graph (Wikipedia of 2012) we will use a later, incremental version of the input graph (Wikipedia of 2013). The size of the graphs used are summarized in Table 1.

The smallest graph we use has 89,000 nodes and the largest 17 million. Their ratio of positive:negative edges goes from 1:11,000 in the best case (Wordnet) to 1:27 million in the worse case (Wikipedia). This huge imbalance makes of LP a *needle in a haystack* problem, where we are trying to find a tiny set of correct edges within a huge set of incorrect ones. This inconvenient setting must not be considered as something abnormal or to be fixed. Instead we must accept it as an intrinsic property of large and sparse graphs and try to work around it. In the case of LP we deal with the class skew problem by focusing on high certainty link predictions. Low certainty predictions typically entail a large number of mistakes (*i.e.*, a large FPR), which makes LP useless: for graphs the size of the ones we use here, incorrectly accepting a 0.01% of all non-existent edges (FPR of 0.0001) represents more incorrectly accepted edges than the 100% of all positive edges (TPR of 1). If we intend find real domains of application for LP in graphs this size we must reduce the number of miss-classifications. For that reason we consider the overall AUC measure is not appropriate for evaluating the performance impact of LP algorithms in large sparse graphs. More relevant is the left-bottom corner of the ROC curve, where the best TPR/FPR ratios are achieved. The high threshold predictions found in that section of the curve represent the most precise and therefore applicable results. In that regard we have found that LP scores which perform better at high thresholds do so consistently.

Data set source	Number of nodes	Number of edges (Input + Test)
Cyc	116,835	345,599
Wordnet	89,178	698,588
Wikipedia	17,170,894	166,719,367

Table 1. Size of graphs used for evaluation

4.1 Cyc

The Cyc [Lenat, 1995] project was started in 1984, by D. Lenat, with the goal of enabling AI applications with human-like common sense reasoning. In its almost thirty years of existence the Cyc project has developed a large KB containing those facts that define common sense according to its creators. The project uses a declarative language based on first-order logic (FOL) called CycL to formally represent knowledge. OpenCyc is a reduced version of Cyc which contains most of the same concepts, taxonomic relationships, well-formedness information about predicates and natural language strings associated with each term. In 2012 a version of OpenCyc was released which included the entire Cyc ontology in OWL, implementing RDF vocabulary.

We build a semi-formal hierarchy from the OpenCyc's OWL ontology by extracting the `rdfs:Class` elements. These will become the nodes of the graph we try to predict links on. To implement the edges of the graph we use the RDF relations `rdf:type` and `rdfs:subClassOf`, as these define a kind of hierarchy. Consequently, a LP process in this graph will discover new edges of a merged `rdf:type` and `rdfs:subClassOf` kind, among `rdfs:Class` elements. Notice that only the relation `rdfs:subClassOf` is transitive [RDF_Working_Group, 2014], a common property of hierarchical knowledge partially assumed by our algorithm. These two types of relations, `rdf:type` and `rdfs:subClassOf`, account for over the 80% of all relations among `rdfs:Class` entities in the OpenCyc ontology.

The resultant graph obtained from OpenCyc is directed, unlabelled and composed by 116,835 nodes and 345,599 edges. We run the RA, HRA and INF algorithms on it and compare their results. Considering the whole ROC curve, RA outperforms INF and HRA, as INF can only retrieve the 33% of all correct links before accepting all links. RA on the other hand retrieves the 61% of all correct edges in its less demanding setting (a single shared neighbour), while erring in a 10% of all incorrect edges. Due to that fact, RA has a much better AUC measure w.r.t. the full ROC curve than INF. This is consequence of INF being a more demanding score: while RA considers one shared neighbour as minimum evidence of similarity between nodes, INF requires more than that (see Definitions 3 and 4). When the minimum requirements of INF are met for one or more pairs of nodes (at TPR of 0.33 and FPR of 0.0007), INF outperforms RA by a large margin. And it does so continuously for the rest of the ROC curve with a minimum difference in the FPR of 2 orders of magnitude (see Figure 1). To emphasize on the importance of performance at high thresholds, let us remark that a FPR of 0.01 in the Cyc graph represents 136 million misclassified edges, while the evaluation set contains 34,559 edges. Regarding HRA, it is always worse than

INF but it outperforms RA at high certainty inferences, up to a point where it becomes worse and remains so thereafter (see Figure 1).

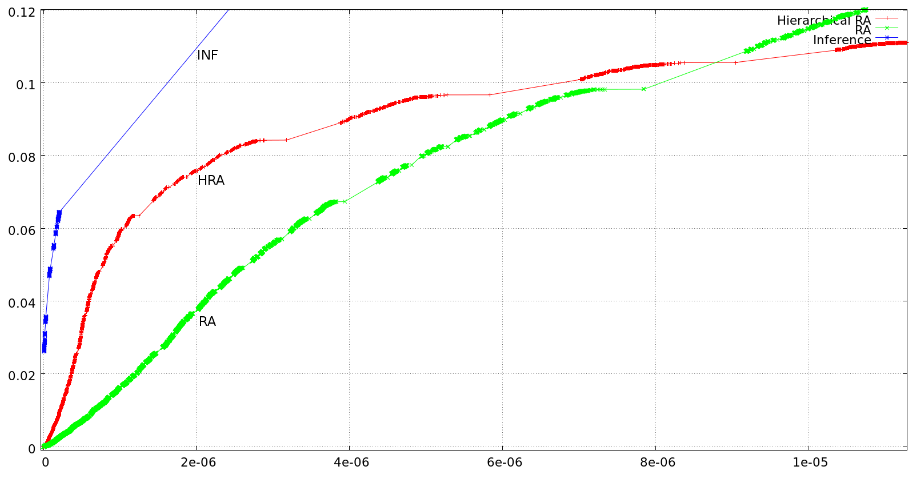

Fig. 1. ROC curve of RA, HRA and INF on the OpenCyc graph for TPRs<12%

Results in the OpenCyc graph indicate that INF is a particularly useful LP local similarity score for predicting hierarchical links in KBs implemented through RDF. The hierarchical assumptions of the INF algorithm match the semantic nature of `rdf:type` and `rdfs:subClassOf` among `rdfs:Class` entities. In that regard, the transitivity property of the `rdfs:subClassOf` relation covers part of the assumptions of the DED score, although DED makes considers transitivity as a weighted measure.

4.2 Wordnet

Wordnet [Miller, 1995] is a lexical database containing words with associated lexical information. In this KB words compose *synsets*, groups of synonym words. Synsets are related to one another through semantic relations such as hypernym/hyponym, meronym/holonym, *etc.* We build the Wordnet directed graph by considering each synset as a node and each *hyponym/hypernym* relation as a directed edge (from hyponym to hypernym). The *hyponym/hypernym* relation in Wordnet is a transitive property. The resultant graph is composed by 89,178 nodes and 698,588 edges. The LP process in this graph will be predicting *hyponym/hypernym* relations among *synsets*.

Wordnet results (see Figure 2) are similar to those of Cyc: RA is better in the AUC measure for the whole ROC curve, but when the minimum requirements of INF are met this score continuously outperforms RA, reducing its FPR up to 6 orders of magnitude. The INF algorithm predicts a 15% of all positive edges without making a single mistake (*i.e.*, TPR of 0.15, FPR of 0). To predict that same 15%, the RA incorrectly accepts 119,289 false positives (*i.e.*, FPR of $1.5e^{-05}$), and the HRA incorrectly accepts 68,710 false positives (*i.e.*, FPR of $9e^{-06}$). HRA once again is worse than INF but outperforms RA at high thresholds, becoming worse than it at a certain point. All three scores perform better in the Wordnet graph than they do in the Cyc graph, probably affected by Wordnet's 3 times smaller sparsity.

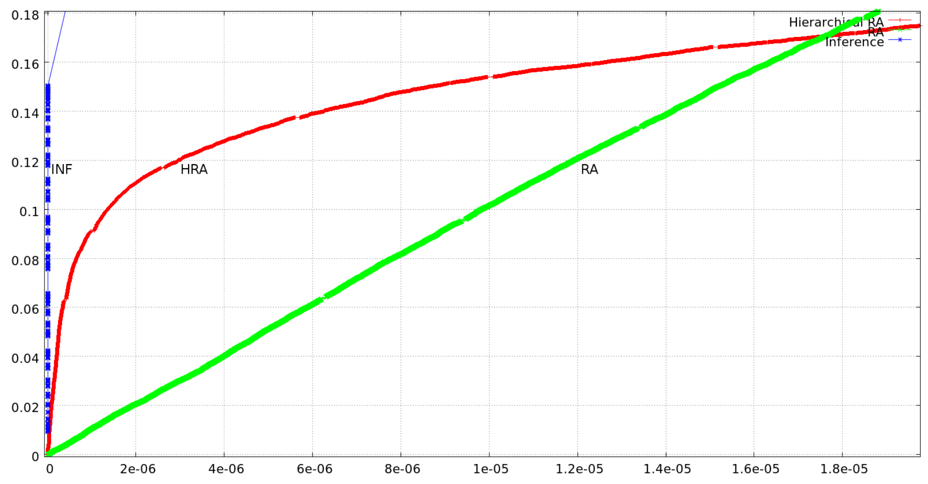

Fig. 2. ROC curve of RA, HRA and INF on the Wordnet graph for TPRs<10%

4.3 Wikipedia

The Wikipedia is a free online encyclopedia containing millions of articles. Articles are linked to one another through *pagelinks*, informal references implemented as hyperlinks. The DBpedia project [Lehmann et al., 2014] periodically extracts data from Wikipedia. We obtained two pagelinks data sets from DBpedia corresponding to the Wikipedia states of June 2012 and June 2013. We build the Wikipedia directed graph using the June 2012 data set, by considering each article as a node and each pagelink as a directed edge. Notice that in this graph, bidirectional relations are frequent (*e.g.*, Spain↔Barcelona). To evaluate LP we use the June 2013 data set as test set. Therefore, we will be evaluating the algorithm capability at predicting real future links in Wikipedia. Links in the 2013 set involving at least one node not found in the input set are removed since they cannot be predicted using only LP methods. The resultant Wikipedia graph has 17,170,894 nodes and 156,028,404 edges. The test set contains 10,690,963 new edges to be predicted. To reduce the cost of exhaustively calculating all possible edges of the graph (294 billion) we focus on those originating from one of the 200,000 nodes with more relations ($|A(x)| + |D(x)|$). We will therefore evaluate over 3.4 billion edges (the equivalent to fully compute a graph with 1.8 million nodes). Of those 3.4 billion edges only 1,804,089 are found in the test set (a 0.000052%). Results of the INF algorithm can be seen in Figure 3.

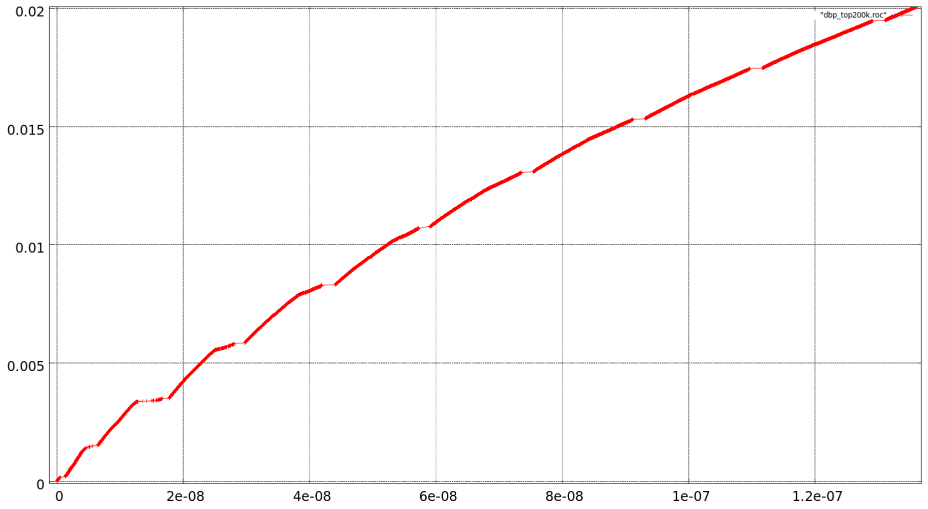

Fig. 3. ROC curve of INF on the Wikipedia graph for TPRs<2%

The Cyc and Wordnet graphs contained a explicit, semi-formal hierarchical structure empowered by transitivity. The Wikipedia pagelink graph however does not. Instead, we hypothesized that such structure was implicitly found in it and that it was relevant for defining its edges, even though in the style guidelines of Wikipedia it says nothing in that regard. The results of the INF algorithm seem to validate our hypothesis as it achieves a TPR of 0.01 (106,909 correctly identified edges) with a FPR of $5.2e^{-8}$ (15M incorrectly identified edges). This rate is better at more demanding thresholds, *e.g.*, a TPR of 0.0001 (1,069 correct edges) with a FPR of $2.5e^{-10}$ (73K wrong edges). The ratio of correct:incorrect predicted edges reaches a minimum difference of one order of magnitude, while the positive:negative class imbalance for this graph is 1:1,903,552 (six orders of magnitude). To further contextualize these results let us remark that, even though all edges not found in the 2013 Wikipedia graph are considered prediction mistakes in this test, many of these edges would be coherent in the Wikipedia. While this test methodology can help us compare the performance of algorithms objectively, it cannot capture the complete potential utility of the results for real domains.

5 Parallelization

Local similarity-based algorithms are the only LP approach currently capable of computing huge graphs. The problem resides in building the whole ROC curve, as for a directed graph we need to evaluate $N * (N - 1)$ different links. In our Wikipedia graph (§4.3) where we tested only links going from the 200,000 most linked nodes to the rest (17,170,894), and the total number of tested edges were already 3.4 billion. Parallelizing this problem is therefore a necessity, even with the simplest of models. As we will see next, LP perfectly fits the paradigm of parallel computing. This notion further reinforces our idea that this type of approach will be key in the future of large graph mining. Next we outline some general features of LP parallelization which may be of interest to the community. All code used in this paper was parallelized using the OpenMP shared memory API [ARB, 2013]. Tests were performed on a single machine with 16 cores running in parallel. In this environment, the most computationally expensive test performed (the Wikipedia graph) finished in 36 hours.

5.1 Embarrassing Parallelization

The task of building the whole ROC curve requires to compute the likelihood of all possible edges. Since each edge can be computed independently from the rest, the task can be parallelized without dependencies. In other words, there are no waiting times between individual link evaluations, which maximizes the use of computational resources. This kind of application is called *embarrassingly parallel*. By calculating the score of all edges we are effectively building the list of all the ROC curve points, each distinct score being a different point. To obtain the position of each point within the curve we need to consider all occurrences of that score within the graph: for each distinct score calculate the total number of true positives (TP) and false positives (FP) obtained by it. Fortunately, this is also an *embarrassingly parallel* problem as each distinct score can be computed independently. In a first parallel block we calculate the score of all edges and their achieved TP and FP. Then in a second parallel block, we build the ROC curve by adding the TP and FP obtained by each score throughout the graph. As a result the process runs efficiently in parallel most of the time.

5.2 Approximation

The size of the LP problem in very large graphs motivates an approximate approach. In the case of the ROC curve we can afford to lose some precision in each edge score without impacting the curve significantly. The number of digits used to represent the score can be of single precision, which are faster to compute and require less storage space than double precision. The resultant ROC curve is virtually indistinguishable from one obtained with double precision digits.

Through the use of single precision digits the number of total points in the final ROC curve diminishes, as points which were very close become the same one. However, for very large graphs, that may still result in an unnecessary large number of points. For example, in the Wikipedia graph (§4.3) our algorithm obtained 11,646,150 points defining the ROC curve. Obviously we do not need so many points to properly represent the curve. We avoid unnecessary points by considering the fact that the ROC curve is a monotonic function (it is entirely non-decreasing). All points aligned between two previously found points *w.r.t.* to one of the two axis can be removed without modifying the overall shape of the ROC curve. As a result we speed up the ROC building process in proportion to the number of points discarded.

5.3 Data Locality

LP is a *data-intensive* tasks as it requires continuous accesses to graph data. When arithmetic operations are simple (as with local scores), data access instructions become the most relevant ones *w.r.t.* algorithm computational efficiency. When data is requested but not found in cache the process must fetch it from main memory (RAM) with the consequent loss of computation cycles. This is specially important for huge data sets that do not fit in memory. To increase data locality and decrease cache misses, we store the graph data sequentially in memory. We use containers that allow us to decide the order in which data is stored, and compute nodes in our algorithm using that same order. This way we maximize data reuse and reduce the number of RAM memory accesses. In our experience, this approach can reduce the computation time by a factor of four.

6 Conclusions

The availability of large graphs motivates research in the field of scalable link mining. The size of graphs originated from the web, social networks or biological relations forces us to use very simple algorithms if those graphs are to be computed in *acceptable* time. At the same time, since these graphs tend to be very sparse LP must achieve a high precision to obtain usable results. As a trade-off between both problems, solutions exploiting inherent low-level properties like the one presented here represent a scalable and rather precise approach to this type of graphs. Advances in this direction can significantly contribute to close the gap between graph mining theory and its practical application.

To compute very large graphs one must find ways of efficiently parallelizing its algorithms. In that regard we have noticed that LP is a very appropriate problem for parallel computing, as it can be divided into independent tasks. This matching makes of LP a field with a promising future, as it will benefit from the ongoing research on parallel computing. This also opens up the possibility of performing LP on even larger graphs than the ones presented here.

Acknowledgements

The experiments have been executed in the MareNostrum supercomputer managed by Barcelona Supercomputing Center (BSC). Authors thank them for their support. This work has been partially supported by the SUPERHUB project (SUstainable and PERsuasive Human Users moBility in future cities - Cofinanced by the European Commission - ICT-FP7-289067). This work has been partially supported by the Catalan Association for Artificial Intelligence (ACIA).

References

[ARB, 2013] ARB, O. (2013). OpenMP Application Program Interface, v.4.0. http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf.

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

[Drumond et al., 2012] Drumond, L., Rendle, S., and Schmidt-Thieme, L. (2012). Predicting rdf triples in incomplete knowledge bases with tensor factorization. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 326–331.

[Franz et al., 2009] Franz, T., Schultz, A., Sizov, S., and Staab, S. (2009). Triplerank: Ranking semantic web data by tensor decomposition. In *The Semantic Web-ISWC 2009*, pages 213–228. Springer.

[Garcia-Gasulla and Cortés, 2014] Garcia-Gasulla, D. and Cortés, U. (2014). Hierarchical Link Prediction. *Submitted to Data Mining and Knowledge Discovery.*

[Getoor and Diehl, 2005] Getoor, L. and Diehl, C. P. (2005). Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12.

[Inokuchi et al., 2000] 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*, pages 13–23. Springer.

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

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

[Lehmann et al., 2014] 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*.

[Lenat, 1995] Lenat, D. B. (1995). CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38:33–38.

[Liben-Nowell and Kleinberg, 2007] Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031.

[Liu and Lü, 2010] Liu, W. and Lü, L. (2010). Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007.

[Lü et al., 2009] 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.

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

[Miller, 1995] Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

[Murata and Moriyasu, 2008] Murata, T. and Moriyasu, S. (2008). Link prediction based on structural properties of online social networks. *New Generation Computing*, 26(3):245–257.

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

[RDF_Working_Group, 2014] RDF_Working_Group (2014). Rdf schema 1.1 (work in progress). Technical report, W3C, http://www.w3.org/TR/2014/PER-rdf-schema-20140109/.

[Zhou et al., 2009] Zhou, T., Lü, L., and Zhang, Y.-C. (2009). Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630.