Reduce and Aggregate: Similarity Ranking in Multi-Categorical Bipartite Graphs

Alessandro Epasto*
Sapienza University of Rome
epasto@di.uniroma1.it

Jon Feldman
Google Research
jonfeld@google.com

Silvio Lattanzi
Google Research
silviol@google.com

Stefano Leonardi†
Sapienza University of Rome
leon@dis.uniroma1.it

Vahab Mirrokni
Google Research
mirrokni@google.com

ABSTRACT

We study the problem of computing similarity rankings in large-scale multi-categorical bipartite graphs, where the two sides of the graph represent actors and items, and the items are partitioned into an arbitrary set of categories. The problem has several real-world applications, including identifying competing advertisers and suggesting related queries in an online advertising system or finding users with similar interests and suggesting content to them. In these settings, we are interested in computing on-the-fly rankings of similar actors, given an actor and an arbitrary subset of categories of interest. Two main challenges arise: First, the bipartite graphs are huge and often lopsided (e.g. the system might receive billions of queries while presenting only millions of advertisers). Second, the sheer number of possible combinations of categories prevents the pre-computation of the results for all of them.

We present a novel algorithmic framework that addresses both issues for the computation of several graph-theoretical similarity measures, including # common neighbors, and Personalized PageRank. We show how to tackle the imbalance in the graphs to speed up the computation and provide efficient real-time algorithms for computing rankings for an arbitrary subset of categories.

Finally, we show experimentally the accuracy of our approach with real-world data, using both public graphs and a very large dataset from Google AdWords.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; E.1 [**Data Structures**]: Graphs and net-works; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

Keywords

Similarity Ranking; Graph Mining; Bipartite Graphs; Personalized PageRank; Random Walks; Markov Chain.

1. INTRODUCTION

Web server log data represents a valuable source of information for understanding global behavior of users. Implicit knowledge extracted from such data can be exploited to better understand and thus satisfy user needs. This practice of *web usage mining* requires the collection of data from various sources, including user profiles, query logs of web search engines, toolbar navigation logs, geo-referenced data, data provided from advertisers and social network data [8, 25, 31, 34, 39] Statistical and data-mining methods (e.g., clustering, classification, ranking, association rules, and sequential pattern discovery) may be employed to detect interesting patterns and connections between users [18, 21, 23].

A fundamental task of web usage mining is to identify users that exhibit common traits, as can be detected from relationships and patterns of interaction with web applications. This problem has several real-world applications, including identifying competing advertisers and suggesting related queries in an online advertising system [8], or finding users with similar interests and suggesting content to them [30]. Graphs provide a universal language to represent relationships between entities and often the different roles of the entities are described by partitioning the nodes of the graphs into different categories. We can represent users/actors on one side of the graph and the items representing data collected from application logs on the other side of the graph. Actors are connected to those items that describe their web usage patterns. Graphs are often labeled on nodes or on edges in order to describe the type and context of the connection or the strength of the interaction.

One major bottleneck to the exploitation of web usage mining is given by the massive amount of data collected even from individual users. As web experience spans most of everyday life, the potential for data collection across different applications, contexts and platforms is almost unlimited. Coping with the heterogeneity and sheer size of this data is a huge challenge for creating more sophisticated personalized services. These graphs are highly unbalanced given the

amount of data we can collect for each individual user. For example, in a graph we constructed out of Google Adwords data, one side (queries) is bigger than the other side (advertisers) by a factor of at least 1000. The challenge in this setting is to perform mining tasks on data with complexity proportional only to the size of the smaller side of the graph, while still preserving the information from the large side that is relevant for one's purposes.

Another important feature is that web usage data are often multi-categorical, as they have been collected from different applications, platforms and contexts. Entities are classified through labels into several categories that are often disjoint (queries, URLs, geo data, opinions, etc.). We are interested in mining relationships between users within in each category; however, it is even more important to mine relationships between users *across* categories in order to personalize services in a more effective way.

Our motivating application is finding related online advertisers. Online advertising campaigns are launched on many market segments. Each market segment is represented by a group of entities (search queries, publisher sites, etc.). Consider for example an advertiser that is targeting the market segments of sport, motors and travel. An industry analyst might like to know the top competitors of that advertiser in each market segment; or, perhaps, on the combination of travel and motors. Further analysis of this graph could also yield insight on entities that the advertiser is missing from their targeting set.

1.1 Our contribution

We study the problem of computing personalized similarity rankings in large-scale multi-categorical bipartite graphs. We represent users and data in a bipartite graph $G(A \cup B, E)$ where A is the set of actors, B is a set of items, E is the set of edges connecting actors to items. The set B of items is partitioned, based on labels, into a set of disjoint categories $\mathcal{C}$. More formally, $\forall b \in B$, $\exists C \in \mathcal{C}$ such that $b \in C$ and $\forall C', C'' \in \mathcal{C}$ if $C' \cap C'' \neq \emptyset$ then $C' = C''$.

Given one actor $a \in A$ and a subset $\mathcal{D} = \{C_1, \dots, C_c\} \subseteq \mathcal{C}$ of categories, our goal is to rank actors of A by similarity to actor a according to subset $\mathcal{D}$. This task corresponds to ranking actor set A by similarity to a in the graph induced by vertex set $\{A \cup C_1 \cup \dots \cup C_c\}$.

We face two main problems. First, the bipartite graph is lopsided, with the item side B bigger by orders of magnitude than the actor side A. It is of crucial importance to reduce the analysis to a subgraph limited by the size of A. Secondly, since $2^{|\mathcal{C}|}$ could be quite large, it is prohibitive to pre-compute a personalized similarity ranking for each actor $a \in A$ and each subset $\mathcal{D} \subseteq \mathcal{C}$. We would like to pre-compute the minimum amount of information needed to answer on the fly a personalized similarity ranking on a specific subset of categories. We address these two challenges as follows.

We provide a novel framework for similarity ranking in multi-categorical data. Our framework is based on the two operators reduce $\odot$ and aggregate $\oplus$. The operator $\odot$ computes a reduced version of the graph induced from nodes $A \cup C_i$ for each category $C_i \in \mathcal{C}$. The size of the resulting graph depends only on the actor set A. The operator aggregate $\oplus$ computes a ranking of actor set A by similarity to $a \in A$ in the graph $A \cup C_1 \cup \dots \cup C_c$; this is achieved via fast aggregation of the information stored in the reduced graph of each individual category. A key property of $\odot$ is that the

reduced graphs computed on the individual categories allow us to answer personalized queries for each actor $a \in A$ and any subset of categories $\mathcal{D} \subseteq \mathcal{C}$.

Our framework is general and versatile. In this work we apply it under several similarity metrics: neighbor intersection, Jaccard coefficient, Adamic-Adar [3], Katz coefficient [17] and Personalized PageRank [5]. This is especially challenging for Personalized PageRank since it requires the use of methods from the fields of stochastic complementation and Markov chain aggregation [26].

Finally, we present extensive experimental results on three datasets: Query-Ads from Google AdWords [2], Authors-Publications from a DBLP dataset [1], Inventors-Patents from a US patent dataset [14]. The reduce operator is able to reduce the size of the graph, in term of number of nodes, by a factor of 2.5 in the smaller DBLP and Patent datasets, up to a factor of > 90 and > 750 in our largest Query-Ads graphs. We evaluate the effectiveness of our similarity rankings in terms of precision and recall against ground truth data from each dataset. In all our datasets, Personalized PageRank and Katz similarity provide a good level of precision/recall. Intersection and Adamic-Adar rank immediately after while the Jaccard coefficient under-performs all the other measures. To evaluate the precision of our Personalized PageRank approximation we compare (on several well-known similarity measures) the values computed from the iterative algorithm to the exact values, showing a fast convergence in few iterations of the iterative algorithm.

2. RELATED WORK

The problem of mining bipartite graphs has received a lot of attention recently. Bipartite graphs can be mined to extract insightful information concerning the relationships between, for instance, keywords and advertisers [8]; queries and websites [25, 39]; words and documents [12]; stocks and financial ratios [32]; gene expression and experiment conditions [24].

Several authors have addressed the problem of identifying relevant nodes in bipartite graphs. Deng et al. [11] introduced Co-HITS, a framework for mining bipartite graphs that incorporates content information on the nodes and that generalizes both HITS [18] and Personalized PageRank [15]. Recently, Wu et al. [39] addressed the problem of integrating click-through bipartite graphs and document features to learn the similarity between documents and queries. Mei et al. [25] employed hitting time based similarity measures to identify related queries from search engine logs. On a related topic, Ng et al. [28] designed a framework, based on the solution of tensor equations, to rank both entities and relations where the relations can belong to multiple types.

Contrary to these works, we focus on similarity functions that can scale to billion-node graphs on parallel systems (like MapReduce [10]), in our multi-category settings. The similarity functions we analyze, which include Adamic-Adar [3], Katz [17] and Personalized PageRank [15], have received attention in the field of information retrieval for several purposes, including predicting link formation [23].

A related topic to similarity in bipartite graphs is co-clustering (a.k.a. bi-clustering); i.e. grouping together nodes on the two sides of the graph that are related. Several authors [12, 13, 27] have tackled this problem, which can be also modeled as an instance of matrix partitioning [4, 12]. In this context, Dhillon [12] introduced a method based

on spectral partitioning for the co-clustering of bipartite graphs representing documents and words. Mirzal and Furukawa [27] addressed the connections between co-clustering in bipartite graphs and clustering in unipartite graphs. Anagnostopoulos et al. [4] provided approximation algorithms and NP-hardness results for some definitions of co-clustering. Another relevant direction in this context is given by the study of heterogeneous networks [41], where the relationships between the entities modeled in the graph are obtained by several different sources.

Finally, another relevant topic to our work is the large body of literature on Markov chain state aggregation [20, 26, 35], jump-started by the work of Simon and Ado [33] on *Nearly Completely Decomposable* chains, which we build on for our Personalized PageRank aggregation algorithm.

Several works have employed such approaches in the context of random walks. Broder et al. [7] applied similar techniques to speed up the computation of PageRank on websites. Parreira et al. [29] applied state aggregation methods to the distributed computation of the PageRank of web pages in a peer-to-peer system where web hosts meet to exchange the results of local computations and to converge to the global scores. Other works focused on updating the PageRank in dynamic graphs [9, 22] and to approximate the PageRank score in local domains [40]. Recently, Vattani et al. [36] studied the closely related problem of preserving the Personalized PageRank score on a subgraph. In our work we apply similar techniques, however their approach cannot be directly applied to our context as their algorithm introduces of a sink node in the reduced subgraph which violates the assumption of disjointness that is necessary in our framework, as we explain later in the paper.

3. COMPUTING SIMILARITY RANKINGS IN BIPARTITE GRAPHS

We present a general approach to computing similarity rankings in bipartite graphs under various similarity metrics. The approach is based on the definition of a reduction operator $\odot$ and an aggregation operator $\oplus$ for each of the measures of interest.

Before starting to describe the technical contribution of the paper we give some useful definition. Let $G = (A \cup B, E)$ be a weighted undirected bipartite graph with non negative edge weights $w : E \to \Re^{\geq 0}$. Denote by $N(a, G) = \{b \in B : (a, b) \in E\}$ the neighbor set of node $a \in A$ in graph G and by $N(b, G) = \{a \in A : (a, b) \in E\}$ the neighbor set of a node $b \in B$ in graph G. We also denote by $\text{sim}_a(b, G)$, $a, b \in A$ a measure of similarity of node b with respect to node a in graph G. The A-node ranking of node a in graph G is obtained by sorting in decreasing order of $\text{sim}_a(\cdot, G)$ the nodes in the A side. We omit graph G when clear from the context.

Given these definitions we can now describe our approach. We recall that the first challenge for our problem is that the huge size of the node set B makes the computation of similarity rankings in $G = (A \cup B, E)$ very expensive. This rules out real-time algorithms, and requires the use of extensive large-scale pre-computation (using MapReduce, for example). We address this issue by showing that it is possible to keep the complexity of the algorithms dependent only on the smaller side A for all the measures of our interest. For this purpose we introduce the reduction operator $\odot$.
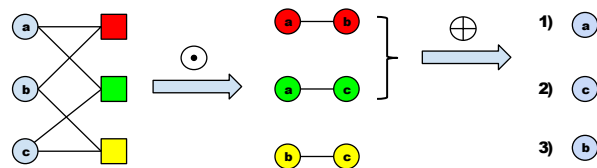

Figure 1: Idealized representation of the framework.

The reduction operator $\odot$ takes as input the entire bipartite graph and produces as output, for each category in isolation, a compact representation of the information needed to compute the similarity rankings in that category. This is obtained by computing a new graph only on the nodes in A for each subgraph $G_i = G[A \cup C_i]$ using large scale computations in MapReduce. We stress that after a phase of preprocessing needed for the implementation of the reduce operator, all other stages of the algorithm will only depend on the size of the set A of nodes.

The problem of designing the operator $\odot$ is formally defined as follows:

Problem 1. Given a bipartite graph $G = (A \cup B, E)$ and a similarity function sim, compute a new weighted graph $\hat{G}(A, E)$ and a new similarity measure sim^* such that, $\forall a, b \in A$, $\text{sim}_a(b, G) = \text{sim}_a^*(b, \hat{G})$.

We use $\hat{G}_i[A \cup C_i]$ to denote the reduced graph, with only A nodes, computed on $G[A \cup C_i]$. We indicate $\hat{G}_i[A \cup C_i]$ by $\hat{G}_i$ when clear from the context. Once we have defined the reduction operator, for each category C_i we can first reduce the graph and then compute the similarity measure between any two nodes.

Our second challenge is the real-time processing of multi-categorical aggregation. This task is problematic because of the number of possible subsets of $\mathcal{C}$ that makes the precomputation of rankings infeasible. For this purpose we define the operator $\oplus$ that uses the information produced by the operator $\odot$, aggregating over a given subset of $\mathcal{C}$ to obtain an exact (or approximate) ranking for a given node $a \in A$. This phase must be implemented by a fast (e.g., few seconds) algorithm. We formally define the aggregation problem:

Problem 2. Given an arbitrary subset of categories $\mathcal{D} = \{C_1', \ldots, C_c'\}$ and reduced graphs $\hat{G}_i[A \cup C_i]$. Let $G' = G[A \cup C_1' \cup \cdots \cup C_c']$ be the induced subgraph of G. For a node $a \in A$, compute efficiently the similarity ranking $\text{sim}_a(\cdot, G')$.

The approach formed by the operators $\odot$ and $\oplus$ is pictorially described in Figure 1, showing the aggregation of the "red" and "green" categories.

In the following subsections we design the two operators that solve Problems 1 and 2 for several measures of node similarity. We present the measures in increasing order of complication, from measures defined on the neighborhoods of nodes to measures involving by the computation of paths between nodes.

3.1 Neighbor intersection

The first notion that we introduce is the *number of common neighbors*. More precisely:

$$\text{INT}_a(b) = |N(a) \cap N(b)|.$$

Reduction. The reduce operator $\odot$ computes $\hat{G}(A, E_A)$, a weighted graph where the weight of an edge between any two nodes $a, b \in A$ is equal to $\mathrm{INT}_a(b, G) = |N(a) \cap N(b)|$ and the similarity measure sim^* between two nodes is equal to the weight of the edge between them, $\mathrm{INT}_a^*(b) = w(a, b)$.

Note that the construction of such a graph requires time in the order of sum of the square of the degree of the nodes in B, $O(\sum_{b \in B} \deg(b)^2)$, which can be quadratic in the worst case for very high degree nodes. This is the same complexity of the reduction phase for the next algorithms. We make two observations. First, in many real graphs, most nodes have low degree and as we observe, the algorithm can scale to very large dataset (billions of nodes in our experiments). Second, the computation time of the intersection can be actually reduced to linear by using well-known min-hashing techniques for set intersection [6] to build the graph and to compute the size of the intersection between any two nodes.

Aggregation. For neighbor intersection the aggregation step is done very efficiently. Since the categories form a partition of node set B, the neighbor intersection on categories $\mathcal{D} = \{C_1', \ldots, C_c'\}$ is equal to the sum of neighbor intersection in each of the categories of $\mathcal{D}$, namely,

$$\mathrm{INT}_a(b, G') = \sum_{C_i \in \mathcal{D}} \mathrm{INT}_a(b, G_i) = \sum_{C_i \in \mathcal{D}} \mathrm{INT}_a^*(b, \hat{G}_i),$$

with $G' = G[A \cup C_1' \cup \ldots \cup C_c']$, $G_i = G[A \cup C_i]$ and $\hat{G}_i$ is the graph obtained by running the reduction operator on G_i.

We finally note that the aggregation operation can be performed easily also after a min-hashing step. In fact the error will accumulate nicely because we are just using the sum.

3.2 Jaccard coefficient

The Jaccard coefficient of node a with b is defined as the ratio between the *intersection* of $N(a)$ with $N(b)$ and the *union* of $N(a)$ with $N(b)$. Formally, for each node $a \in A$, we define the *Jaccard coefficient* of node a with node b as,

$$\mathrm{JAC}_a(b) = \frac{|N(a) \cap N(b)|}{|N(a) \cup N(b)|}.$$

Reduction. In this case the reduce operator $\odot$ computes a graph $\hat{G}(A, E_A)$ with two weights on each edge. In particular for each pair of nodes a, b we define two weights $w^\cap(a, b) = |N(a) \cap N(b)|$ and $w^\cup(a, b) = |N(a) \cup N(b)|$. We can now define the new similarity measure $\mathrm{Jac}_a^*(b) = \frac{w^\cap(a,b)}{w^\cup(a,b)}$ The time complexity of this task is similar to the reduction step of neighbor intersection, also in this case it is possible to obtain linear time algorithm using well-know min-hash techniques [6].

Aggregation. Using the fact that B is a partition, the aggregation operator on categories $\mathcal{D} = \{C_1', \ldots, C_c'\}$ for the Jaccard coefficient similarity of node a is implemented as follows.

Recall that $G_i = G[A \cup C_i]$ and that $\hat{G}_i$ is the graph obtained by running the reduction operator on G_i. In the first step for a given node a we sum the weights $w^\cap$ of the node a edges in the various categories: $w^\cap(a, b) = \sum_{\hat{G}_i} w_{\hat{G}_i}^\cap(a, b)$, where by $w_{\hat{G}_i}(\cdot, \cdot)$ we denote the weight function $w(\cdot, \cdot)$ between two nodes in the graph $\hat{G}_i$. Similarly $w^\cup(a, b) = \sum_{\hat{G}_i} w_{\hat{G}_i}^\cup(a, b)$.

Then we can compute the Jaccard similarity by simply dividing the two weights $\mathrm{Jac}_a(b, G') = \frac{w^\cap(a,b)}{w^\cup(a,b)}$. The correctness of this step follow from the fact that B is a partition.

3.3 Adamic-Adar

The *Adamic-Adar* similarity [3] metric is defined as

$$\mathrm{AA}_a(b) = \sum_{x \in N(a) \cap N(b)} \frac{1}{\log |N(x)|}.$$

Adamic-Adar differs from intersection since the weight of a common neighbor is equal to the inverse of the logarithm of the degree.

Reduction. The reduce operator for Adamic-Adar is very similar to the reduction operator for neighbor intersection. In this case as well we generate a weighted graph $\hat{G}(A, E_A)$, where the weight of an edge between any two nodes in $a, b \in A$ is equal to $\mathrm{AA}_a(b, G) = \sum_{c \in N(a) \cap N(b)} 1/\log |N(c)|$ and the similarity measure sim^* between two nodes is equal to the weight of the edge between them, $\mathrm{AA}_a^*(b) = w(a, b)$. Note that for Adamic-Adar as well it is possible to implement this step in linear time using a weighted permutation and standard hash-min techniques as in [6].

Aggregation. For the aggregation step it is easy to observe that given that categories are disjoint, we can just sum up the scores, $\mathrm{AA}_a(b, G') = \sum_{C_i \in \mathcal{D}} \mathrm{AA}_a^*(b, \hat{G}_i)$.

Notice that also for this measure the aggregation operation can be performed easily after a min-hashing step.

3.4 Katz

The KATZ_β measure [17], for $\beta \in (0, 1)$, of a with respect to b is defined as

$$\mathrm{KATZ}_{\beta,a}(b) = \sum_{l=1}^{\infty} \beta^l |\mathcal{P}_{\mathcal{G}}(a, b, l)|.$$

with $\mathcal{P}_{\mathcal{G}}(a, b, l)$ defined as the set of distinct paths of length exactly l between a and b in graph G.

Reduction. The reduction operator for KATZ_β is slightly more complex than the previous operators. We start by noticing that the length of any path between two nodes in A is always even because the graph is bipartite.

This simple observation suggests that we can shrink the length two paths in single edges, rigorously we build the weighted graph $\hat{G}(A, E_A)$ where the weight between two nodes a, b is equal to the number of 2-steps paths between them $w(a, b) = |N(a) \cap N(b)|$. To design the similarity function KATZ_β^*, we note that every even path can be split in a sequence of length two paths. Furthermore to compute the multiplicity of even paths of a specific length l it is enough to consider all the possible sequence of $l/2$ 2-steps paths between them and multiply their multiplicities.

Using this observation and the fact that the length of a path between two nodes in A is always even we can define the new similarity measure as:

$$\mathrm{KATZ}_{\beta,a}^*(b) = \sum_{i=1}^{\infty} \beta^{2i} \left(\sum_{p \in \mathcal{P}_{\hat{G}}(a,b,i)} \left(\prod_{e \in p} w(e) \right) \right).$$

Aggregation. The aggregation operator for the KATZ_β similarity is impractical because it involves the computation of paths of possibly infinite length. For this reason we need to approximate the KATZ_β score considering path up to a specific length.

In Section 4.4 we show experimentally that considering paths up to length 4 gives already a good approximation of the KATZ$_\beta$ similarity measure. So in this section we focus on aggregating paths of maximum length 4.

Recall that we define $G' = G[A \cup C'_1 \cup \ldots \cup C'_c]$, $G_i = G[A \cup C_i]$ and that $\hat{G}_i$ is the graph obtained by running the reduction operator on G_i. Unfortunately in this case it is not enough to simply aggregate the scores computed in the categorical graphs $\hat{G}_i$ because those similarity scores take into account only the paths within a single category but not the paths across categories. Notice that since length 2 paths lies inside a single category, this is an issue only for paths of length 4. We can however compose those paths by a first length 2 path within a category C'_i and a second length 2 path within a different category C'_j.

Given these considerations we split the length 4 paths in intra-category paths and inter-category paths. For the former we can use the same technique of the reduction step, for the latter we have to pay attention to combine length 2 paths from different categories. More rigorously, we have that

$$
\begin{aligned}
\text{KATZ}^*_{\beta,a}(b) &= \beta^2 \sum_{C_i \in \mathcal{D}} w_{\hat{G}_i}(a,b) \\
&+ \beta^4 \sum_{C_i \in \mathcal{D}} \left(\sum_{p \in \mathcal{P}_{\hat{G}_i}(a,b,2)} \left(\prod_{e \in p} w(e) \right) \right) \\
&+ \beta^4 \sum_{\substack{C_i, C_j \in \mathcal{D}, \\ C_j \neq C_i}} \left(\sum_{c \in N(a) \cup N(b)} w_{\hat{G}_i}(a,c) w_{\hat{G}_j}(c,b) \right),
\end{aligned}
$$

where in the first line we consider all the path of length 2 between a and b, in the second line we consider all the intra-category paths and in the third line we consider all the inter-category paths. This computation takes at most $O(\deg(a)|\mathcal{D}|^2)$ time where $|\mathcal{D}|$ is the number of categories aggregated.

3.5 Personalized PageRank

Finally, we introduce the Personalized PageRank (henceforth PPR) similarity measure for node $a \in A$ [15].

Definition 1. Let $G = (V, E)$ be a weighted graph and let $a \in V$, $\alpha \in (0,1)$. $\overrightarrow{\text{PPR}}(G, \alpha, a)$ is defined as the vector presenting the stationary distribution of the following random walk on G. The walk starts in node a. At each step, if the walk is in node x, with probability α it *jumps* to node a, otherwise it moves to $y \in N(x)$ with the *ordinary random walk* transition probability $p(x,y) = \frac{w(x,y)}{\sum_{z \in N(x)} w(x,z)}$.

Notice that we will use the same notation when $G = (A \cup B, E)$ is bipartite. Notice also that, in contrast to the previous measures, PPR naturally takes into account weights of the edges.

Now we can define the Personalized PageRank similarity for the pair of nodes $a, b \in A$ as $\text{PPR}_{\alpha,a}(b) = \overrightarrow{\text{PPR}}(G, \alpha, a)(b)$.

In the rest of the section we concentrate on the challenges of defining the two operators for PPR rankings.

Reduction. The reduction operator for the Personalized PageRank is a bit more complex than in the other cases. First we build the weighted graph $\hat{G} = (A, E_A)$ where the

weights[1] of the edges E_A are defined as follows $\forall x, y \in A$:

$$
w_A(x,y) = \sum_{z \in N(x) \cap N(y)} \frac{w(x,z) w(z,y)}{\sum_{u \in N(z)} w(z,u)}.
$$

Notice that for $\forall x, y \in A$ the probability of going from a to b in a 1-step ordinary random walk on $\hat{G}$ is the same of performing a 2-step walk on graph G between them.

Then, to define the similarity measure PPR*, we first introduce the following Lemma, of which we only sketch the proof in this paper.

Lemma 1. Let $\hat{G} = (A, E_A)$ be the weighted graph with nodes A and weights w_A, then

$$
\overrightarrow{\text{PPR}}(G, \alpha, a)[A] = \frac{1}{2 - \alpha} \overrightarrow{\text{PPR}}(\hat{G}, 2\alpha - \alpha^2, a),
$$

where $\overrightarrow{\text{PPR}}(G, \alpha, a)[A]$ is the subvector of $\overrightarrow{\text{PPR}}(G, \alpha, a)$ containing only the probabilities of nodes in A.

PROOF. *(Sketch):* The ordinary random walks between A nodes, i.e. without jumps, can be simulated efficiently by looking at the 2-step random walk transition probabilities matrix. The PPR walk is however complicated by the presence of jumps to node a. In particular, at first sight it is not clear how to capture the fact that a random walk may restart while visiting a node in B. The definition of stationary distribution helps us, as intuitively we need only to compute the fraction of time that the random walk spend in each node. For this it is enough to capture the probability that in a 2-step PPR walk we do not jump, which is $(1 - \alpha)^2$. Conversely, the probability of restarting is precisely $2\alpha - \alpha^2$. Hence, the PPR stationary distribution conditioned on being in a node in A is $\text{PPR}(\hat{G}, 2\alpha - \alpha^2, a)$ and to get the correct distribution we can simply multiply this distribution with the probability of being in the A side, which can be proved to be always $\frac{1}{2-\alpha}$ in any bipartite graph, irrespectively of the topology.

These intuitions can be formalized using the theory developed by Meyer in [26]. $\square$

Using the above Lemma we can now define PPR* as

$$
\text{PPR}^*_{\alpha,a} = \frac{1}{2 - \alpha} \overrightarrow{\text{PPR}}(\hat{G}, 2\alpha - \alpha^2, a).
$$

Notice that an additional positive by-product of the reduction is that each step of a walk in $\hat{G}$ represents two steps in G so any power iteration algorithm will converge twice as fast.

Aggregation. Similar to the previous measures, we can apply the $\odot$ operator, on the subgraph $G_i = G[A \cup C_i]$, for each subset $C_i \in \mathcal{C}$, to produce a graph $\hat{G}_i$ from which we can compute efficiently the PPR similarity between nodes. We now show how, based on such graphs we can aggregate the PPR rankings in each category to compute in an efficient way the PPR similarity on the subgraph $G[A \cup C'_1 \cup \ldots \cup C'_c]$.

We need a few additional information on the structure of the bipartite graph to proceed in the definition of the $\oplus$ operator.

Definition 2. For a any given $x \in A$ and subset $C_i \in \mathcal{C}$, let us define $U_x(C_i) = \sum_{y \in N(x) \cap C_i} w(x,y)$.

---

[1] The transition matrix on $\hat{G} = (A, E_A)$ is also known hidden transition matrix in other works [11].

Definition 3. For a any given $x \in A$ and subset $C_i \in \mathcal{C}$, let $F_x(C_i, C_j)$ be the probability of reaching any node in C_j, after performing a 3-step standard random walk, in the bipartite graph $G(A \cup B, E)$, starting from the node x and conditioned to the fact that the first step ends in a node in C_i.

Notice that both information can be efficiently precomputed in MapReduce with at most 3 MapReductions. So, in our real-time $\bigoplus$ operator we assume to possess such values.

For our definition of the $\bigoplus$ operator we build on the iterative aggregation-disaggregation algorithm of Koury et al. [20].

The main idea behind this algorithm is the following. Consider a Markov chain whose states are partitioned in a family of disjoint sets $S_1, S_2, \ldots S_c$ and suppose to have an initial approximation $\bar{\pi}$ for the stationary distribution. Let $\bar{\pi}_i$ be the subvector of $\bar{\pi}$ with only values in S_i and consider the $c \times c$ transition matrix T between subsets, i.e. the matrix were T_{ij} is the probability of moving between S_i and S_j at stationary.

Koury et al. [20] show that based on an approximation of the stochastic matrix T we can obtain an improved approximation of the actual stationary distribution of the system by a linear combination of the vectors $\bar{\pi}_i$ whose weights are based on the stationary distribution of T.

This operation can be repeated arbitrary many times[2] and under certain assumptions on the Markov chain described in details in [20] the algorithm converges in the limit to the exact stationary distribution.

While a very powerful technique, there are a few key algorithmic challenges that must be overcome to apply this method to our problem. First of all, in order to approximate the ranking on the subgraph $G[A \cup C'_1 \cup \ldots \cup C'_c]$, we need to aggregate the stationary distributions on subsets of nodes $G_1 = G[A \cup C'_1], \ldots, G_c = G[A \cup C'_c]$ that are not disjoint. So we cannot apply directly the results in the Markov chain state aggregation theory [26] as they rely on the disjointness of the sets of states aggregated. Second, we want to implement the $\bigoplus$ operator as a real-time algorithm, which means that all the computations depending on the actual classes $C'_1, \ldots, C'_c$ aggregated, which are only known at the run-time, must be as efficient as possible.

We address both issues and adapt the algorithm of Koury et al. to solve our problem by relying on a series of results that we postpone at the end of this section for sake of the presentation. We proceed now by defining the algorithm for the operator $\bigoplus$.

The input of the $\bigoplus$ operator is the node a for which we want to compute the PPR ranking and the set of the stationary distributions[3] $\bar{\pi}_i = \overrightarrow{\text{PPR}}(\hat{G}_i, 2\alpha - \alpha^2, a)$ on the graphs $\hat{G}_i$ obtained by applying the operator $\bigodot$ to the graph $G_i = G[A \cup C'_i]$, for each $C'_i \in \mathcal{D} = \{C'_1, \ldots, C'_c\}$. The algorithm has access to the precomputed $F_\cdot(\cdot, \cdot)$ and $U_\cdot(\cdot)$ values and to the adjacency lists of the reduced graphs $\hat{G}_i$ for $C'_i \in \mathcal{D}$.

Each iteration of the operator $\bigoplus$ proceeds as follows:

- Compute the $c \times c$ transition matrix T:

$$
\begin{aligned}
T_{ij} = &(2\alpha - \alpha^2)\frac{U_a(C'_j)}{\sum_{C'_k \in \mathcal{D}} U_a(C'_k)} + \\
&(1 - \alpha)^2 \sum_{x \in A} \bar{\pi}_i(x)\frac{F_x(C'_i, C'_j)}{\sum_{C'_k \in \mathcal{D}} F_x(C'_i, C'_k)}.
\end{aligned}
$$

- Compute the stationary distribution $t = (t_1, \ldots, t_c)$ of the matrix T.

- For each $C'_j \in \mathcal{D}$ compute,

$$
\begin{aligned}
\phi_j &= \sum_{i=1}^{c} t_i \sum_{x \in A} \bar{\pi}_i(x) U_x^{-1}(C'_j)\frac{\hat{G}_j(x)}{\hat{G}_j(x)e}, \quad (1) \\
\hat{\pi}_j &= (2\alpha - \alpha^2)\mathbf{1}_a + (1 - \alpha)^2 \frac{\phi_j}{\phi_j e}, \quad (2)
\end{aligned}
$$

where $\mathbf{1}_a$ is a vector with all zeros except a 1 in position a (the size is assumed by the context), $\hat{G}_j(x)$ is the vector representing the row of node x in the adjacency matrix of the reduced graph $\hat{G}_j$ and e is a vector with all 1s.

At the end of each iteration, $\hat{\pi}_i$ is fed to the algorithm as the next $\bar{\pi}_i$ vector, and the process is repeated until a convergence criteria is met or the maximum number of steps is reached. Finally, in the last step of the algorithm, instead of computing Equations 1 and 2, we use the last $\bar{\pi}_i$ vectors to compute the approximation $\hat{\pi}$ of the stationary distribution we are interested in,

$$
\begin{aligned}
\phi &= \sum_{i=1}^{c} t_i \sum_{x \in A} \bar{\pi}_i(x) \left(\sum_{j=1}^{c} U_x(C'_j)\right)^{-1} \sum_{j=1}^{c} U_x(C'_j)\frac{\hat{G}_j(x)}{\hat{G}_j(x)e}, \\
\hat{\pi} &= \alpha\mathbf{1}_a + \frac{(1 - \alpha)^2}{2 - \alpha}\phi.
\end{aligned}
$$

The following theorem, whose proof is omitted from this paper, shows an important property of the $\bigoplus$ algorithm.

Theorem 1. Under the assumptions of Koury et al. algorithm [20], for $t \to \infty$ number of steps of the PPR operator $\bigoplus$, the value $\hat{\pi}$ converges to the distribution on nodes in A in the graph $G' = G[A \cup C'_1 \cup \ldots \cup C'_c]$ i.e. $\overrightarrow{\text{PPR}}(G', \alpha, a)[A]$.

The complexity of each step of the algorithm is approximately given by the total length of the rankings aggregated in input, times the average degree of the nodes in the reduced graphs plus the time to compute (or approximate) the stationary distribution on a small $c \times c$ matrix.

Sketch of the proof.

The rest of the section is devoted to provide a sketch of the rather lengthy proof of the theorem.

As already said our main issue is the fact that the subgraphs we want to aggregate are not disjoint as they all contain the nodes in A.

Similarly to the result in Lemma 1, we can show however that the process can be also reduced efficiently to a graph with only nodes in B whose weights depend on the 2-step ordinary random walk transition probabilities between nodes in B. In this case the reduction operation is slightly complicated by the fact that the restart probability is on a node that is actually not present in the subgraph we restrict to.

---

[2] After applying at end of each step some simple matrix operations to avoid the algorithm to be trapped in fixed point.
[3] We assume that a has at least an edge in each C'_i category aggregated, otherwise such distributions would be trivial with all probability in a.

Graph	$|A|$	$|B|$	$|E|$	$|E_A|$
DBLP	881,759	1,295,405	3,773,586	6,277,745
Patent	1,496,067	2,139,313	4,301,229	4,220,151
Q-A-Cost	$> 1 \times 10^6$	$> 100 \times 10^6$	$> 150 \times 10^6$	$< 50 \times 10^6$
Q-A-Impr.	$> 1 \times 10^6$	$> 1.5 \times 10^9$	$> 5 \times 10^9$	$< 1.5 \times 10^9$

Table 1: Properties of the graphs analysed. Column $|E_A|$ refers to the number of edges in the reduced graph with only A nodes. The exact figures for the proprietary Query-Ads graphs are not given.

Lemma 2. Consider the graph $\hat{G}_B = (B, E_B)$, with weights given by $\forall x, y \in B$, $w_B(x, y) = \sum_{z \in |N(x) \cap N(y)|} \frac{w(x,z)w(z,y)}{\sum_{u \in N(z)} w(z,u)}$, then

$$\overrightarrow{\mathrm{PPR}}(G, \alpha, a)[B] = \frac{1 - \alpha}{2 - \alpha} \sum_{b \in N(a)} p(a, b) \, \overrightarrow{\mathrm{PPR}}(\hat{G}_B, 2\alpha - \alpha^2, b),$$

where $\overrightarrow{\mathrm{PPR}}(G, \alpha, a)[B]$ is the subvector of $\overrightarrow{\mathrm{PPR}}(G, \alpha, a)$ containing only the PPR stationary on nodes in B and $p(a, b)$ is the ordinary random walk transition probability between a and b in the bipartite graph G.

The proof of this lemma, which is omitted, depends again on the method of stochastic complementation for Markov chains, surveyed by Mayer in [26].

From the previous Lemma we know that we can define a Markov chain containing only nodes in B and where the categories of nodes we want to aggregate are disjoint sets, so we can apply the algorithm of Koury et al. This would constitute a correct solution of our problem; let us call this algorithm "Naive $\bigoplus$".

The use of Naive $\bigoplus$ poses two new challenges. First, we reduced our Markov chain to the larger side of the graph, which is very inefficient. Second, the reduced graph on nodes B for graph $G[A \cup C'_1 \cup \ldots \cup C'_c]$ depends on the actual categories aggregated and it cannot be neither computed in the real-time operator $\bigoplus$ nor we can pre-compute each possible reduced graphs for each subset of $\mathcal{C}$. For this reason, we would like to be able to compose, on-the-fly, the information obtained in the individual category's reduced graphs, possibly considering only information related to nodes in A.

It turned out that both objectives can be actually achieved by using an important property of PPR on bipartite graphs: that the stationary distribution on either side of the graph uniquely determines the one on the other. More precisely, the following lemma holds.

Lemma 3. Consider any weighted bipartite graph $G(A \cup B, E)$. Let π_A and π_B be the stationary distributions of PPR on the two sides of the graph, i.e. $\pi_A = \overrightarrow{\mathrm{PPR}}(G, \alpha, a)[A]$, $\pi_B = \overrightarrow{\mathrm{PPR}}(G, \alpha, a)[B]$. Then

$$\pi_A = \alpha \mathbf{1}_a + (1 - \alpha)\pi_B \bar{W}, \qquad \pi_B = (1 - \alpha)\pi_A W,$$

where W is the $|A| \times |B|$ transition matrix of the ordinary random walk from states in A to states in B and conversely, $\bar{W}$ is the opposite transition matrix (from states in B to states in A).

First, it is possible to notice that by using Lemma 3 we can easily compute, for instance with a single MapReduce step, the distribution on the B side from the information on the reduced graph. In our motivating example, that means that we can also define similarity (or better relatedness) of

items for a given user. Second, by tackling the bijective relationship between the stationary distributions on A and B, we can redefine Naive $\bigoplus$ to operate only on the distributions on the A side.

Using this intuition, by some algebraic manipulation it is possible to show that the graph $\hat{G}$ obtained by operator $\odot$ on $G[A \cup C'_1 \cup \ldots \cup C'_c]$, and containing only nodes A, can be reconstructed from the individual graphs $\hat{G}_i$ reduced from $G[A \cup C'_i]$, which means that the information precomputed are sufficient to define our operator $\bigoplus$.

4. EXPERIMENTAL RESULTS

Section 4.1 describes our datasets; Section 4.2 reports the effect of the reduce procedure on the graphs analyzed; Section 4.3 presents our results on the accuracy of our method with respect to ground truth data; Section 4.5 gives an empirical evaluation of the approximation error of our PPR aggregation algorithm.

4.1 Datasets

Our analysis concerned several private and public large-scale dataset as reported in Table 1.

Query-Ads. Our largest datasets, *Query-Ads (Cost) and Query-Ads (Impression)*, which we denote as Q-A, are two proprietary graphs obtained by data collected from Google anonymized query logs where the identity of the advertisers is anonymized and the queries have been hashed and can only be identified by their hash id. Nodes in these weighted bipartite graphs represent advertisers (A set) and queries (B set) in the Google AdWords [2] system. The advertisers are connected to the queries on which their ads have been shown. We consider two variants of this dataset: one where edge weights measure the cost paid by the advertiser, and the other where edge weights count the number of times the advertiser appears (has an *impression*). Nodes in the B side are partitioned in 24 disjoint categories each representing a different market segment.

DBLP. The *DBLP* graph is a publicly available [1] snapshot of a publication graph in computer science. Nodes in this (unweighted) bipartite graph are authors (A set) and publications (B set), and each author is connected to her publications. Publication nodes belong to 6162 different venues that can be partitioned in categories according to their field. In our experiments, for simplicity, we consider three categories for this graph: (1) top tier web-related and data mining conferences (WWW, KDD and WSDM); (2) top theoretical computer science conferences (STOC, FOCS, SODA and ICALP); (3) the rest of the papers. To get a glimpse of results produced by our approach, which we evaluate thoroughly in Section 4.3, we show in Table 2 the first few positions induced by PPR for a well-known author in this dataset. The table shows both the authors with the highest similarity for the reference author in the theory and data mining community and the ranking obtained by the aggregation of both.

Patent. The *Patent* graph is obtained by a public dataset [14] containing about 2 million U.S. patents granted between January 1963 and December 1999 and their relative citations received between 1975 and 1999. In this bipartite graph, nodes representing the inventors (A set) are connected with unweighted edges to the patents (B set) in which they are listed as authors. The patents are classified by the U.S.

#	Theory	WWW-like	Both
1	E. Tardos	L. Backstrom	E. Tardos
2	D. Kempe	J. Leskovec	L. Backstrom
3	A. Kumar	D. P. Huttenlocher	J. Leskovec
4	P. Raghavan	R. Kumar	D. P. Huttenlocher
5	M. Sandler	J. Ugander	D. Kempe

Table 2: Example PPR rankings for the prolific author Jon Kleinberg in the two categories evaluated.

Patent Office in 36 disjoint categories each describing the field of the invention (for instance "Computer Hardware & Software", "Biotechnology", etc).

4.2 Graph Reduction

As discussed in the introduction, one of our objectives is to design algorithms that can be applied efficiently to bipartite graphs with a significant imbalance in the sizes of their two sides. While the actual definition of the reduce operation $\odot$ changes from measure to measure, they share a commonality: they all define edges between pairs of A nodes that have a common B neighbor. Thus the number of edges in the reduced graph is always the same, while the weights may be different. Table 1 shows statistics on the effect of the reduction procedure in our dataset. Notice that we always observe a significant reduction in the number of nodes: this ranges from a factor ~ 2.5 in DBLP and Patent, up to a factor of > 90 and > 750 in our largest graphs Query-Ads (Cost) and Query-Ads (Impression), respectively. This reduction plays a fundamental role in enabling the scalability of ranking computations to the large scale graphs in our Query-Ads datasets. Notice how the number of edges is also significantly reduced in our Query-Ads dataset (to $< 1/3$ of the original size) while left substantially unaltered in the Patent dataset. The DBLP dataset shows instead a relative increase in the number of edges, probably because the input graph is extremely sparse.

4.3 Ranking evaluation

In this section we evaluate the ability of the various methods analyzed to provide meaningful rankings in our datasets. As the semantics of the relationships represented in the graphs varies across our datasets, this evaluation is necessarily data-dependent.

Our general approach works as follows: for each of our datasets we derive ground truth clusters of nodes identified to be relevant for a given node; then we measure the ability of the methods to find the the nodes in the ground truth cluster associated to that node (and hopefully rank them high as well). To do so we employ two well-known metrics in the field of information retrieval: *precision* and *recall*. For a given ranking, the precision at position x of the ranking is defined as the fraction of nodes in the top x positions that are in the ground truth set. Conversely, the recall at position x measures the ratio between the number of nodes in the ground truth set among the top x positions and the size of the ground truth set itself. The precision and recall values at different positions of the ranking induce a precision vs recall curve, which can be used to compare the accuracy of different ranking methods. We proceed by describing the source of ground truth information used in our analysis for each of our datasets.

Query-Ads. For the Query-Ads graphs we employed ground truth information gathered by the Google AdWords team.

This proprietary dataset contains, for each advertiser in a sample of ~ 1000 advertisers, a set of most similar advertisers.

DBLP. In the case of DBLP graph, as a proxy for the similarity between authors, we use a well-known and simple natural language processing technique, the n-gram similarity [16]. More precisely, for each author, after removing stopwords from the titles of her papers, we induce the set of bi-grams (i.e. sequence of 2 consecutive words) of all her titles. For a given parameter k we use as ground truth the top k authors in DBLP in terms of matching bi-grams (ties broken randomly). To make our analysis more significant we consider only authors with at least 5 papers in the bipartite graph used in this experiment.

Patent. For the patent graph we use as the source of ground truth the co-citations [38] among patents[4]. More precisely, for two inventors a and b, we define $s(a, b)$ as the number of patents citing both a patent filed by inventor a and a patent filed by inventor b. Similarly to the DBLP graph, the ground truth set of inventor a is defined by the top k inventors in decreasing order of $s(a, b)$ (ties broken randomly). In this case as well we restrict to inventors with at least 5 patents.

While we acknowledge that both the bi-gram and the co-citation measures have a positive bias towards direct co-authors (co-inventors) of a given node, we stress that neither the citations nor the titles play any role in the definition of the bipartite graph given in input to the algorithms.

Rankings in weighted graphs.

Contrary to the other datasets, the Query-Ads graphs are weighted. Note that while the PPR random walk method has a natural definition for weighted graphs, such a generalization is not so obvious for the other measures (for instance the Adamic-Adar or Jaccard coefficient). Given the importance played by weights in this dataset we consider in our analysis both the weighted version of PPR and a simple generalization of the intersection measure.

For each node $a \in A$, we define as the *weighted intersection* of node a with node b the following similarity measure $s_a(b) = \sum_{x \in N(a) \cup N(b)} w(b, x)$.

In other words, the similarity $s_a(b)$ for given pair $a, b \in A$ of nodes is given by the sum of the weights of the edges connecting node b to the common neighbors with a in the bipartite graph. Notice that this similarity degenerates to intersection if all edges have unit weight. The A-node ranking of node a is obtained by sorting in decreasing order of $s_a(\cdot)$ the nodes in the A side.
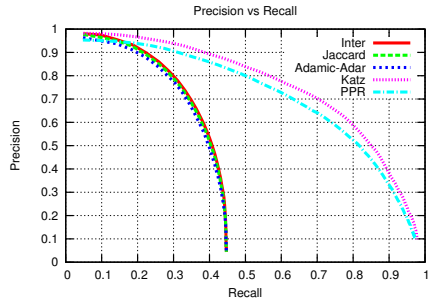
4.4 Results

Figure 2 shows the average Precision vs Recall curves on a sample of nodes. Results for Query-Ads are averaged over the ~ 1000 nodes for which ground truth information is available. For DBLP and Patent we set the parameter k to 20 and average over a random sample of 1000 nodes with a least k nodes in the ground truth cluster.

In all our datasets, both the weighted and unweighted version of PPR and the Katz similarity provide a good level of performance, thus justifying their increased complexity. Intersection and Adamic-Adar rank immediately after and show very similar results, while the Jaccard coefficient un-
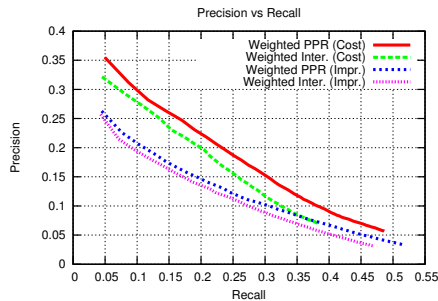
---

[4]Citation data appears to be less noisy than title bi-grams but it is unavailable in our DBLP dataset.
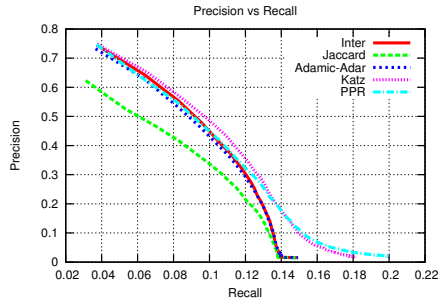
(a) **DBLP**

(b) **Patent**

(c) **Query-Ads**

Figure 2: Precision vs Recall curves at various positions of the rankings. We use $k = 20$ nodes in the ground truth clusters. PPR and Katz outperforms the other algorithms in every dataset. Results are averaged over ~ 1000 nodes for each graph.
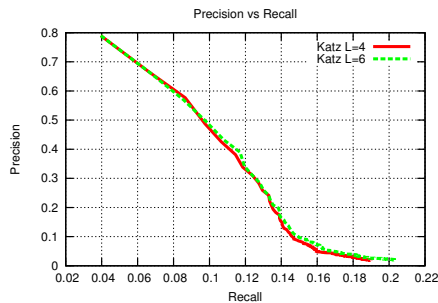

Figure 3: Precision vs Recall in DBLP for the Katz measure depending on the maximum length L considered. No significant improvement is shown when using longer paths. Similar results are observed for Patent. We averaged over 100 nodes using $k = 20$.

derperforms all the other measures, in particular for the DBLP dataset. We notice that the overall recall achieved is lower in our DBLP dataset, probably due to the noisy ground-truth used (title bi-gram similarity).

Note that the computation of the *exact* PPR is infeasible for the scale of our dataset, as it requires a matrix inversion of size $|A| \times |A|$. Consequently, we compute the PPR distribution using the approximation method of Andersen et al. [5], setting $\alpha = 0.15$ and the approximation parameter to $\epsilon = 0.0001$, unless otherwise specified. Similarly, in the computation of the Katz ranking we consider only paths of length up to 4 and we use $\beta = 0.05$. Figure 3 shows no significant improvement in precision using longer paths, while the computation becomes very expensive for paths of length 8 or more.

4.5 PPR aggregation algorithm

In this section we evaluate experimentally the precision with which PPR is approximated in the iterative aggregation algorithm introduced in Section 3.5.

For each of our datasets, we performed the following experiment: we execute the reduce algorithm on the entire bipartite graph. Then we apply our iterative aggregation algorithm, for a sample of 1000 nodes in A and a certain number of steps, to aggregate the PPR ranking of two categories. Then, to evaluate the correctness of PPR, we compare the results with PPR computed on the subgraph with the categories aggregated. Both for the *ground truth* ranking and for the individual category ranking, we use the previously mentioned approximation algorithm with $\epsilon = 0.0001$.

To evaluate the precision of our approximation we employ several well-known similarity measures: the Kendall's tau correlation index [19], the cosine similarity and the Pearson correlation coefficient; which we now recall.

The Kendall's tau correlation index is a well-known ranking agreement measure. The index ranges between -1 to $+1$ and measures the prevalence of pairs of elements that have the same order in both rankings. An index of $+1$ shows perfect concordance (i.e. the rankings are equal), while -1 indicates instead a total disagreement (i.e. one ranking is the opposite of the other). A value close to 0 characterizes rankings that are a random permutation of each other.

More precisely, we employ the following definition of the Kendall's tau index [19] which accounts for the presence of ties, arising for instance for nodes with zero PPR:

$$\tau = \frac{C - D}{\sqrt{(C + D + F)(C + D + S)}},$$

where C and D are the number of concordant and discordant pairs, respectively; F and S are the number of ties only in first and only in the second ranking respectively. Ties occurring in both rankings are disregarded.

The Kendall's tau assesses the agreement over the entire range of positions. To measure the precision of the rankings for the first few positions (the ones most likely to be seen in many applications), we compute as well the index τ restricted to the first k positions of the ranking. This is obtained by considering only pairs of elements in the top k positions of the ground truth ranking.

Figure 4 shows the results for Kendall's tau index after running one iteration of our algorithm, for both the entire ranking and the first positions. The results show a very strong positive agreement between the correct ranking and
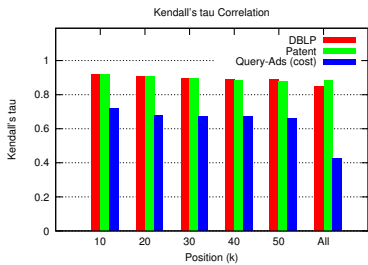
Figure 4: Average Kendall's tau correlation between the ground truth PPR **ranking and our approximation after one iteration of the aggregation algorithm.**
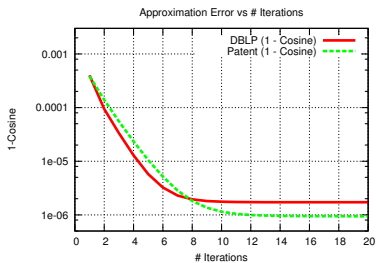

Figure 5: Variation of the approximation error depending on the number of iterations executed. Lines for the Pearson coefficient are practically coincident.

our approximation in all the graphs, with higher values in our DBLP and Patent datasets. As expected, the agreement is stronger in the top positions which are less subject to additive approximation errors due to their larger PPR score.

The Kendall's tau correlation measures the similarity of the order in the two rankings, but it does not assess the accuracy in approximating the probability values. For this reason, we further evaluate the accuracy of our PPR approximation by applying two vector similarity measures on the distributions. Let a and b be two vectors of n elements. We define the following well-known similarity measures: the cosine similarity and Pearson correlation coefficient.

The cosine similarity is defined as follows:

$$Cosine(a,b) = \frac{\sqrt{\sum_{i=1}^{n} (a_i b_i)}}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}}.$$

Cosine similarity measures the angles between the two vectors and it ranges, in non-negative vectors, from 0 (orthogonal vectors) to 1 (same direction).

Similarly, the Pearson correlation coefficient between a and b is defined as

$$Pearson(a,b) = \frac{\sqrt{\sum_{i=1}^{n} \left((a_i - \bar{a})(b_i - \bar{b})\right)}}{\sqrt{\sum_{i=1}^{n} (a_i - \bar{a})^2} \sqrt{\sum_{i=1}^{n} (b_i - \bar{b})^2}},$$

where $\bar{a}$ and $\bar{b}$ indicates the average value in a and b, respectively. This measure assesses the linear dependency of the two vectors and it ranges from -1 to $+1$ where 1 shows a perfect positive correlation, 0 shows no correlation and -1 indicates a perfectly negative correlation.

Using the previous definitions we can now compare the ground truth PPR distribution with the one obtained by our

Graph	Cosine Sim.	Pearson Coeff.
DBLP	$0.9996 \ (\pm 4.7 \times 10^{-5})$	$0.9996 \ (\pm 4.8 \times 10^{-5})$
Patent	$0.9995 \ (\pm 7.2 \times 10^{-5})$	$0.9993 \ (\pm 1.2 \times 10^{-4})$
Q-A-Cost	$0.9716 \ (\pm 3.3 \times 10^{-3})$	$0.9698 \ (\pm 3.5 \times 10^{-3})$

Table 3: Similarity between the ground truth PPR **ranking and our approximation after one iteration of the aggregation algorithm. Results between parentheses are the** 95% **confidence intervals, obtained by the t-student distribution [37].**

iterative method. Table 3 shows the accuracy of the rankings obtained after a single step of the iterative algorithm in our datasets. In all our datasets the results confirm the conclusion suggested by the Kendall's tau measure: we observe a very high accuracy after a single iteration, showed by Cosine similarity and Pearson correlation coefficient very close to 1. The approximation error is again particularly small in our smaller datasets: DBLP, Patent.

Finally, Figure 5 shows the evolution of the approximation error as the number of iterations increases. In this experiment, we determine the ground truth PPR distribution using a very small ϵ value ($\epsilon = 1 \times 10^{-6}$) on our smaller DBLP and Patent dataset. The initial category rankings used in the aggregation are still however computed using $\epsilon = 1 \times 10^{-4}$ as in the other experiments. Notice that the approximation error decreases steeply in the first steps and then reaches a plateau consistent with the approximation error of the ground truth PPR after about 8-10 iterations.

5. CONCLUSIONS

Bipartite graphs representing the relationships between actors and items in online services can be mined to extract many useful insights.

In this work we studied the problem of efficiently computing similarity rankings in massive bipartite graphs where the items can be partitioned in arbitrary subsets. We introduced a novel algorithmic framework that enables the real-time computation of several widely-used similarity measures in large-scale graphs. These algorithms, crucially, tackle the lopsided nature of such graphs to execute the computation on the small side of the network (which can reduce the number of nodes by a factor of > 750 in our experiments).

We provide both a thorough experimental evaluation of the accuracy of our framework for large-scale publicly available and proprietary datasets, and a formal proof of the correctness of the algorithms.

We believe that our approach can be extended to include several other similarity measures developed in the literature, for example the HITS [18] algorithm. It would also be interesting to extend the framework to more nuanced approaches that integrate additional non-topological information about the entities in the graph (e.g., [11]). Another important and challenging open problem consists in generalizing our approach to the practically relevant case where the categories are not disjoint.

Acknowledgements

6. REFERENCES

[1] DBLP dataset (accessed on 12 Sept 2013). http://dblp.uni-trier.de/xml/.

[2] Google AdWords. www.google.com/adwords.

[3] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social networks*, 2003.

[4] A. Anagnostopoulos, A. Dasgupta, and R. Kumar. Approximation algorithms for co-clustering. In *PODS*, 2008.

[5] R. Andersen, F. Chung, and K. Lang. Using PageRank to locally partition a graph. *Internet Mathematics*, 2007.

[6] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, 1998.

[7] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient PageRank approximation via graph aggregation. *Information Retrieval*, 2006.

[8] J. Carrasco, D. Fain, K. Lang, and L. Zhukov. Clustering of bipartite advertiser-keyword graph. In *ICDM*, 2003.

[9] S. Chien, C. Dwork, R. Kumar, D. R. Simon, and D. Sivakumar. Link evolution: Analysis and algorithms. *Internet Mathematics*, 2004.

[10] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 2008.

[11] H. Deng, M. R. Lyu, and I. King. A generalized Co-HITS algorithm and its application to bipartite graphs. In *KDD*, 2009.

[12] Dhillon and S. Inderjit. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, 2001.

[13] D. Greene and P. Cunningham. Spectral co-clustering for dynamic bipartite graphs. In *DyNaK: Dynamic Networks and Knowledge Discovery*, 2010.

[14] B. H. Hall, A. B. Jaffe, and M. Trajtenberg. The NBER patent citation data file: Lessons, insights and methodological tools. Technical report, National Bureau of Economic Research, 2001.

[15] T. H. Haveliwala. Topic-sensitive PageRank. In *WWW*, 2002.

[16] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, first edition, 2000.

[17] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 1953.

[18] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 1999.

[19] W. R. Knight. A computer method for calculating Kendall's tau with ungrouped data. *Journal of the American Statistical Association*, 1966.

[20] J. Koury, D. McAllister, and W. J. Stewart. Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains. *SIAM Journal on Algebraic Discrete Methods*, 1984.

[21] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. *Computer networks*, 1999.

[22] A. N. Langville and C. D. Meyer. Updating Markov chains with an eye on Google's PageRank. *SIAM Journal on Matrix Analysis and Applications*, 2006.

[23] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 2007.

[24] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2004.

[25] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, 2008.

[26] C. D. Meyer. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM review*, 1989.

[27] A. Mirzal and M. Furukawa. Eigenvectors for clustering: Unipartite, Bipartite, and Directed Graph Cases. In *ICEIE*, 2010.

[28] M. K.-P. Ng, X. Li, and Y. Ye. MultiRank: co-ranking for objects and relations in multi-relational data. In *KDD*, 2011.

[29] J. X. Parreira, C. Castillo, D. Donato, S. Michel, and G. Weikum. The juxtaposed approximate PageRank method for robust PageRank approximation in a Peer-to-Peer web search network. *The VLDB Journal*, 2008.

[30] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[31] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 1999.

[32] K. Sim, J. Li, V. Gopalkrishnan, and G. Liu. Mining maximal quasi-bicliques to co-cluster stocks and financial ratios for value investment. In *ICDM*, 2006.

[33] H. A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica: Journal of The Econometric Society*, 1961.

[34] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*, 2000.

[35] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, 1994.

[36] A. Vattani, D. Chakrabarti, and M. Gurevich. Preserving personalized PageRank in subgraphs. In *ICML*, 2011.

[37] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye. *Probability and statistics for engineers and scientists.* Prentice Hall., 1993.

[38] H. D. White and K. W. McCain. Bibliometrics. *Annual review of information science and technology*, 1989.

[39] W. Wu, H. Li, and J. Xu. Learning query and document similarities from click-through bipartite graph with metadata. In *WSDM*, 2013.

[40] Y. Wu and L. Raschid. ApproxRank: Estimating rank for a subgraph. In *ICDE*, 2009.

[41] D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles. Co-ranking authors and documents in a heterogeneous network. In *ICDM*, 2007.