Ranking Marginal Influencers in a Target-labeled Network

Pranay Lohia
IBM Research AI, India

Kalapriya Kannan
IBM Research AI, India

Karan Rai
Microsoft, United States

Srikanta Bedathur
IIT Delhi, India

ABSTRACT

Using social networks for spreading marketing information is a commonly used strategy to help in quick adoption of innovations, retention of customers and for improving brand awareness. In many settings, the set of entities in the network who must be the targets of such an information spread are already known, either implicitly or explicitly. It would still be beneficial to route the information to them through a carefully chosen set of influencers in the network. We term networks where we have such vertices labeled as targeted recipients as *targeted networks*. For instance, in an online marketing channel of a fashion product, where vertices are tagged with 'fashion' as their preferred choice of online shopping, forms a targeted network. In such targeted networks, how to select a small subset of vertices that maximizes the influence over target nodes while simultaneously minimizing the non-target nodes which get the information (e.g., to reduce their spam, or in some cases, due to costs)? We term this as the problem of maximizing the marginal influence over target networks and propose an iterative algorithm to solve this problem. We present the results of our experiment with large information networks, derived from English Wikipedia graph, which show that the proposed algorithm effectively identifies influential nodes that help reach pages identified through queries/topics. Qualitative analysis of our results shows that we can generate a semantically meaningful ranking of query-specific influential nodes.

ACM Reference Format:
Pranay Lohia, Kalapriya Kannan, Karan Rai, and Srikanta Bedathur. 2020. Ranking Marginal Influencers in a Target-labeled Network. In *7th ACM IKDD CoDS and 25th COMAD (CoDS COMAD 2020), January 5–7, 2020, Hyderabad, India.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3371158.3371197

1 INTRODUCTION

Social networks are increasingly useful in several activities ranging from e-commerce [2, 10] to job recruitments [3]. Tangible benefits of using social networks are achieved only when we are able to identify the appropriate set of nodes in the social network, be it *influential nodes* or *high-centrality nodes*.

In this paper, we consider a crucial variant of the problem of identifying and ranking of nodes which have the potential to influence a large social network. We observe that in many settings, there are certain nodes which can be labeled upfront as target nodes and the goal of a campaign is to ensure these nodes labeled as *targets* are reached with sufficiently strong paths. In other words, the goal is not to find globally influential nodes in the network, but find nodes that maximize the ability to reach and influence these targeted nodes in the network. At the same time, the problem also has the implicit requirement that non-target nodes should not be activated in order to minimize their spam or the cost of communication involved. We motivate this with a scenario from recruitment network where we first encountered this problem.

Recruitment Network Scenario: *Consider a concrete scenario of a recruiter who would like to identify certain set of nodes in a professional network. These nodes can recommend a job advertisement to members in their network who fit the job requirements (i.e., target nodes). Clearly the recruiter would not like this advertisement to reach candidates who do not qualify (i.e., non-target nodes). From the recruiters perspective, it will increase the effectiveness as only those nodes that match the job description will respond. This will also avoid "spamming" the other candidates who do not match the requirements. Also, if many non-relevant job descriptions are sent to a candidate, there is a possibility of the recruiter being tagged as a "spammer". This can lead to a scenario where a potentially perfectly matching job description is ignored by the candidate. We observe that existing traditional approaches of selecting influencers do not address these issues. Figure 1 shows an example scenario where the central node 'Query node' represents the recruiter. M represents the level of matching to the recruiters query say searching for candidates with matching skill set for a job description. M having zero indicates non-matching nodes and a real values indicate the level of matching the the query. Thus green node have full match, while the yellow nodes match according to M. The goal is to avoid reaching red nodes that does not have any match while we aim to reach the matching nodes at any level.* □

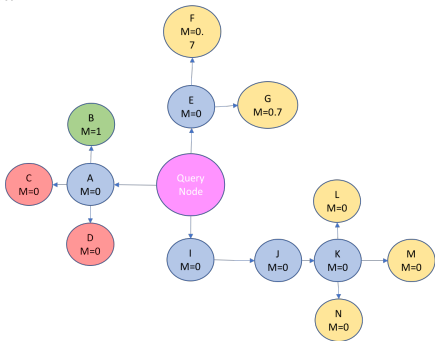

Figure 1: Example graph illustrating different nodes

Our goal is identifying nodes that not only have the ability to reach a large number of *target* nodes that should be recipients of the information but also minimize the number of undesirable recipients of the information at the same time. We model this as a problem of ranking nodes based on their *marginal influence* score. Intuitively, a node has a high marginal influence score if it can

CoDS COMAD 2020, January 5–7, 2020, Hyderabad, India

ACM ISBN 978-1-4503-7738-6/20/01…$15.00
https://doi.org/10.1145/3371158.3371197

influence many valuable target nodes and, at the same time, reaches minimal number of low value nodes in the network.

Interestingly, application of our marginal influence ranking on a Wikipedia-like setting exhibits an unique feature. In a Wikipedia graph, for each search string, it helps in listing those pages that can lead to the target pages and avoid broad categories of matches, thus providing better way to navigate. We have implemented our algorithm for a recruitment solution. Feedback obtained suggests that our algorithm has higher effect compared to the traditional referral sourcing in the business setting.

Our contribution in the paper are as follows: (1) We introduce the notion of marginal influencers (MI) and propose an iterative algorithm to rank nodes by *benefit* and *loss* in terms of them influencing target nodes. (2) Evaluation results over a large Wikipedia corpus and enterprise recruitment application.

2 RELATED WORK

Finding top influencer(s) have been a subject of large interest to research community. References [7] and [4] talk about general influence maximization and how it can be made more efficient. In general influence maximization, one tries to select the best set of seed nodes such that the information can be sent to as many nodes as possible. There is no notion of quality of nodes or even their properties. References [14], [11] and [16] look at a different aspect of a similar problem. Their objective is to find best matching documents with respect to a query node such that tagging of those documents are relevant to the user. In a sense, here they are considering the influence of other network users for deciding which are the best documents for someone to look at.

Finding MI on labeled nodes is tackled in [8] where they look at influencing nodes with specific labels. But they do not consider how many non-labeled nodes have been influenced. or if they should have any effect on the actual influence scores. In [18] and [9], they look at trying to maximize positive influence while removing users who will propagate a negative opinion of the information. For this, they use diffusion models and since it is an NP-hard problem, they use approximation methods. In contrast, we consider the net effect of benefits considering a node's neighbors benefits.

PageRank is the basic core concept that looks at ranking web pages based on their influencing capacity in an iterative fashion in [12]. It's a very fundamental idea that takes into account only the links between nodes. In [6], they take this idea forward for calculating the inverse influence between nodes but this is almost entirely directed towards the idea of link prediction. Hence, they calculate them between pair of nodes only. We leverage the concept of PageRank to address the problem through an iteration process, however we address a different problem.

On a separate note, [15] show a method on how to include query node bias into normal Wikipedia query search. PageRank and Katz Centrality measures both are calculated in a similar iterative fashion. But both these models only incorporate the edges of the graph and connectivity in the network. They do not take into consideration, the labelling or matching (in our case) of attributes that is connected with each node. They do not provide appropriate results for our problem set-up. They have the same time complexity as our iterative model, $o(m + n)$ where m is the number of nodes and n is the number of edges in the graph. Closest to our work is the work presented in [13]. However, in [13] the target was to obtain an objective function to maximize targeted influence for which they have used an information diffusion based technique. Our focus is on coming up with a scalable and efficient algorithm. Finally, [13] shows identification of target vs non-target nodes are submodular and can be addressed independently. In their work they have considered the notion of time of diffusion information to derive nodes that are influenced over time. In our work we use the propagation quantity and aggregate the influences to consider the net benefit and do not discount the benefits due to time factor.

3 MODELING MARGINAL INFLUENCERS

Marginal influence score of a node measures the relative influence of the information originating from that node on the population of target nodes versus the non-target nodes. We model this process using its *benefit* and *loss* functions. In this section, we develop this model formally.

3.1 Problem Setting

We operate on an edge-weighted, directed graph $G = (V, E, w)$, where V is the vertex-set, E is the edge-set and $w : E \rightarrow \mathbb{R}_{\geq 0}$ is a function that associates a real-valued weight with each edge to represent the strength of relationship between the two nodes connected by the edge. In real-world networks, we treat this as a probability measure. It models the probability of the propagation by the source node, given information along that edge. A special vertex $Q \in V$ is designated as the query vertex responsible for initiating the information flow. The nodes which are targeted for activation (either by Q as the target audience, or by prior preference settings of nodes) are modelled through a non-negative weight function $y : V \rightarrow \mathbb{R}_{\geq 0}$, that corresponds to the relevance of a vertex to the information which is being propagated from the vertex Q. In concrete settings, this weight could correspond to a matching score of the profile of a node v_i to a job description expressed as a set of keywords.

We are now equipped to formally define the process of information flow on the network, and thus formulate the problem of maximizing the marginal influence.

Definition: [Marginal Influence Score] Given an edge-weighted graph $G = (V, E, w)$, a query vertex Q, a subset $A := \{v \in V \mid y(v) > 0\}$, then the marginal influence score of a node u is defined as:

$$\sum_{\substack{v_i \in A, \\ Q \xrightarrow[u]{\star} v_i}} W(Q \xrightarrow[u]{\star} v_i) - \sum_{\substack{v'_j \in V \setminus A, \\ Q \xrightarrow[u]{\star} v'_i}} W'(Q \xrightarrow[u]{\star} v'_j),$$

where $W(Q \xrightarrow[u]{\star} v)$ is a function that determines the value of information flow from(to) node Q to(from) node v through the vertex u – aggregated over all paths that connect these nodes in the graph. We distinguish by W and W' between the cases when v is in the target set or otherwise.

For simplicity of exposition, in the rest of the paper, we use the term *benefit score* to indicate $W(\cdot)$ and *loss score* for $W'(\cdot)$, and model them appropriately. Now, the overall problem of identifying the best k nodes which collectively maximize the marginal influence is defined as:

Definition: [Marginal Influence Maximization] Given an edge-weighted graph $G = (V, E, w)$, a query vertex Q, a subset $A := \{v \in V \mid y(v) > 0\}$, then the goal is to select a k-size subset $I_k \subset V \setminus (A \cup Q)$ such that,

$$\underset{I_k}{\text{argmax}} \sum_{u \in I_k} M_u,$$

where M_u is the marginal influence score of a node u.

3.2 Modeling the Value of Information Flow

To compute the Marginal Influence Scores, we first model the **benefit score** –i.e., the influence score of a node derived from the target nodes in the network– and the **loss score** –i.e., the influence score of a node derived from the non-target nodes, independently. We adopt an iterative process similar to Katz centrality [17], that propagates benefit (or loss) scores until the values converge. We compute both scores independently and the overall marginal influence score of a node is measured as the difference between its benefit and loss scores.

The benefit score associated with a node measures the influence that the node has w.r.t. the target nodes. Let B_i denote the benefit score of a node v_i. The equation for computing the benefit score of a vertex v_i is given by,

$$\begin{aligned} B_i^{(k+1)} &= \lambda \frac{\delta(i)}{|V|} + (1-\lambda) \sum_{(v_i,v_j)\in E} \frac{w(v_i,v_j)}{|E|} \left(\alpha B_j^{(k)} + (1-\alpha) y_j\right), \\ \delta(i) &= \begin{cases} 1, & \text{if } v_i \text{ is the querynode} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

where λ and α are decay and mixture-model parameters respectively, which can be empirically set (or learnt). The first component in the above equation biases the scores to prefer nodes that are closer to the query node. The higher the λ value, the more preference given to the nodes closer to the query node. Similarly α models the weight that is given to the benefit score of the neighbors as opposed to their matching scores, in computing their benefit scores. The second component of the Equation 1, models the benefit score of the nodes as a function of the benefit scores of its neighbors from the previous round (i.e., k-th round) and its neighbors matching score. The B_i values in each round are normalized by the total number of nodes in the graph so that $\sum_{v_i \in V} B_i = 1$.

The above equation can be used iteratively, starting with weights on all nodes, $y(v_i)$ – possibly based on their match to a query, until we obtain a computational convergence.

Next, we will move onto defining a loss score (i.e., $W'(\cdot)$) for a node. Similar to the benefit score computation, we compute loss scores – denoted as L_i for a vertex v_i using the following equation, $\mathrm{L}_i^{(k+1)} = \sum_{(v_i,v_j)\in E} \frac{w(v_i,v_j)}{|E|} \left(\beta L_j^{(k)} + (1-\beta)(1-y_j)\right)$,where β models how much weight is to be given to the loss score of the neighbors as opposed to their weight. According to the Equation 3.2, the loss captures the inverse of benefit scores, which is reflected in $(1 - y_j)$. Like benefit score propagation, the loss score propagation to the neighbors is also determined by the weight of the node. A normalized value of $w(i, j)$ is used to do it. The benefit and loss scores are computed simultaneously for the network at each iteration, however the number of iterations that benefit scores and the loss scores converges upon might vary with each input query string and also the size of the graph. The main difference between the benefit score and the loss score is that benefit scores account for presence of high benefit and matching nodes while loss scores account for presence of low benefit and non matching nodes.

Algorithm 1 ComputeBLScores(G) - Computing B and L scores

Require: Network G with nodes and edges, Query String S, Query node **return** Top ranking Marginal Influential nodes

```
initialization - B_i^(0) = L_i^(0) = 0 for all nodes, k = 0
Max_score = 0
for v_i ← V do
    M_i = LuceneIndex(i,Query)
    if M_i > Max_score then
        Max_score = M_i
    end if
end for
for v_i ← V do
    M_i = M_i / Max_score
end for
error = 0
while error < 10^-10 do
    k++
    for v_i ← V do
        B_i^(k) = 0
        L_i^(k) = 0
        for v_j ← neighbours of v_i do
            B_i^(k) += (1 − λ) * (w(v_i,v_j)/|E|) (α B_j^(k−1) + (1 − α) M_j)
            L_i^(k) += (w(v_i,v_j)/|E|) (β L_j^(k−1) + (1 − β)(1 − M_j))
        end for
    end for
    B_q^(k) += λ/|V|
    error = 0
    for i ← V do
        error+ = (B_i^(k) − B_i^(k−1))^2 + (L_i^(k) − L_i^(k−1))^2
    end for
    error = error^0.5
end while
```

3.3 Marginal Influencer Computation

As mentioned in the previous section, the values of the benefit and loss scores said to have converged when their values reach a steady state. However, in practice, since we are interested in the overall marginal influence score, we define the convergence as the situation when their RMS value of their differences in successive rounds of computation have reached as stable state. In other words,

$$Error_{RMS} = \sqrt{\sum_{v_i \in V} \left(B_i^{(k)} - B_i^{(k-1)}\right)^2 + \left(L_i^{(k)} - L_i^{(k-1)}\right)^2} \leq \Delta \tag{2}$$

Algorithm 1 presents the algorithm that takes as input the graph G and computes the benefit and loss scores for all nodes in an iteration. It is assumed that $w(v_i, v_j)$ on edges represents the strength of connection in a probabilistic sense, and the scores $y(v_i)$ on (target)nodes derived by executing a keyword query on the text associated with each vertex – we denote it as M_i to highlight this. For instance, in the recruitment network the query string is the job description and the resume information consisting of skills, experience, work profile etc., form the node information. We assume that this information is structured and for unstructured text we fall back on well known indexers (Lucene [1] here) to compute the matching scores. In line 10, the values of Matching scores computed for each nodes are normalized so that they are within the range $[0, 1.0]$. The iterations continue until the error value is less than

the threshold value of $\Delta = 10^{-10}$ (terminating condition in line 13). Lines 16-23 compute B_i and L_i for all v_i nodes in V. At the end of each iteration, lines 26-29 compute the error between the previous iteration and the current iteration. The implementation specific details of the algorithm are given in Section 5. The vales of α, β and λ are empirically chosen over through multiple runs over different graphs. Some of the factors we identified are size of graph, nature of graph (connected, strongly connected, tree etc.) that affect these values.

3.4 Selection of Budgeted set

We explore two settings to select the top-k MI's. In one setting, we identify top-k MI's which will maximize strength the of influence to the set of best matching nodes despite same nodes being targetted. In another setting, it is desired to identify MI's who can reach distinct set of matching nodes. We refer to the former case *Maximizing Strength of Influence (MSI)* as latter case as *Maximizing Targets for Influence (MTI)*

3.4.1 Maximizing strength of influence (MSI) Computation. Consider a recruitment network, in which there are a set of matching nodes whose matching scores are high. A recruiter would like to identify set of nodes that will increase the probability of the job description reaching the matching nodes. It is likely that these matching nodes can be reached through several MI's (a property that will hold for connected and strongly connected graphs). We would like to identify the set of MI's (top m) nodes from the list of MI's (obtained accordingly to Algorithm 1) for each such set of matching nodes. Let the set of matching nodes identified by Q be $SN_1 ... SN_n$, in which each $SN_i \in i$, represent a sub-set of nodes from the set of matching nodes. For each SN_i, it is desired to maximize the possibility of reaching the nodes i.e. we would like to reach the matching nodes in more than one path possible. This increases the chances of information reaching them.

We devise an algorithm that utilizes the sub-modularity property described earlier in this section to identify the set of nodes that can obtain the MSI's. The algorithm is iterative that efficiently removes the identified MI in the previous iteration and recomputes a new MI for the set of matching nodes described. The algorithm described in Algorithm 2 illustrates this in detail. By removing the *MI* already selected, we provide an opportunity to reach the same matching nodes via other paths. The iterations are repeated either until all the matching nodes are covered or the budget has been exhausted.

Algorithm 2 presents the details of the computation of MSI nodes. Steps 2, initializes the set of nodes for which MSI's need to be identified. This is either a node or set of nodes that are picked up from matching nodes. The nodes in one set can overlap with nodes in another set. Step 4-14 computes the MSI's iteratively. Step 5, computes the B and L scores using the Algorithm 1. From the values obtained, $IdentifyTopMarginalInfluencer$ returns the node with the highest $B - L$ score. This is added to the output list Step 7. This node is removed from the graph (the node, in-degree edges to this and out-degree edges from this node) and this new graph is considered for the next iteration. The effect of this node on the network is completely removed. The iterations are repeated until either the budget (number of required node) is met or all the input subsets of matching nodes are visited.

Algorithm 2 Deriving MSI nodes

Require: Network $G(V, E)$, Query , Query node S, Matching nodes Set SN, Budget for each Set B_{s_i} for each set in SN
return Set of MSI
Set $SN = SN_{1....n} \subset MS$ ▷ Subsets of nodes from the Matching Set MS
2: **for** $SN_i \in SN$ **do**
for $i \leftarrow 1$ to B_{s_i} **do**
4: G_{bl} =ComputeBLScores(G) ▷ G_{bl} represents the graph with the computed B and L for each node
x= IdentifyTopMarginalInfluencer(G_{bl})
6: addToOutput(x)
Remove x from $V \in G$
8: **for** $y|(x, y) \in E$ **do**
Remove (x,y) from E;
10: **end for**
for $y|(y, x) \in E$ **do**
12: Remove (y,x) from E;
end for
14: **end for**
end for

3.4.2 Maximizing Targets of influence (MTI) computation. Consider a scenario of a recruitment network where the recruiter prefers to identify different matching nodes, in terms of number of matching nodes. In this problem, it is desired to maximize the reach of influence. In other words, we want to reach as many matching nodes as possible. It is likely that the *MI* picked by algorithm proposed in Section 3.3 can return MI that leads to the same set of matching nodes. We consider a budget is given to the query node. Our aim is determine from the set of MI's, a set referred to as MTI who will lead to different set of matching nodes.

We leverage the sub-modularity explained earlier to iteratively identify the nodes by varying the graph structure. In this case, once we select an influential node, we remove all matching nodes that it allows us to reach. Then, we iteratively execute the algorithm to find the next node that can allow reachability to another set of targets. The iteration is continued till the budget is reached.

Algorithm 3 Identifying node for maximum reachability

Require: Network G with nodes and edges, Query, Query node, Budget
return Set of influential nodes for maximum reachability (MTI)
Set $S = null$
List $L = null$
for $i \leftarrow 1$ to B_{s_i} **do**
2: G_{bl} = computeBLScores(G);
x = IdentifyTopInfluence(G_{bl})
4: addToOutput(x)
Add x to L
6: **while** L is non-empty **do**
$y \leftarrow$ First element of L
8: Remove y from E
for $z|(y, z) \in E$ **do**
10: Remove (y,z) from E
Add z to L
12: **end for**
end while
14: **end for**

Algorithm 3 presents the algorithm for computing the nodes that can influence maximum number of targeted/matching nodes (MTI) given a budget in terms of number of MTI's to select. As in the previous case of identifying MSI's we use an iterative approach, but this time we eliminate all the target nodes that are possible to reach through the identified MTI. We begin with identifying the top marginal influencer (again using the algorithm presented in Section 3.3) - Step 3 and 4. We add this top MI to the output list in Step 5. For this top MI, we identify and remove all matching nodes (since we wanted to find new set of MI that can reach to different set of targets). Steps 7-12 removes the matching nodes,

Algorithm 4 Illustration of algorithm for sensitivity analysis

Require: Network G with nodes and edges, Query, Query node, Algorithm 1 **return** Most sensitive edge
BestEdge=-1
OriginalScore=0
G_{bl}= ComputeBLScores(G)
I = ComputeInfluencerList(G_{bl})
for $k|k \in I$ **do**
    OriginalScore=OriginalScore+B[k]-L[k]
end for
BestScore=OriginalScore
for $j|(i,j) \in E$ **do**
    if $P(i,j) > 0.9$ **then**
        Continue
    end if
    Score=0
    $P(i,j) = P(i,j) + \theta$ to get Modified Network G_1
    G_{bl}= ComputeBLScore(G_1);
    I_1 = ComputeInfluencerList(G_{bl})
    for $k|k \in I_1$ **do**
        Score=Score+B[k]-L[k]
    end for
    if $Score > BestScore$ **then**
        BestScore=Score
        BestEdge=(i,j)
    end if
end for

its in-coming edges and its out-going edges. The graph is iterated again to obtain new set of MTI until the budget is exhausted.

4 SENSITIVITY ANALYSIS

One of the important uses of this model is to provide the benefit of not only identifying the marginal influential nodes but also studying the network for connection sensitivity. This helps the query node to analyze the connection properties for obtaining matching nodes. Sensitivity analysis answers one or more of the following questions: (a) For a given query string, what are important connections that need to be improved and (b) For a given query string, what is the optimum level of connection strength required to reach matching nodes. This requires sensitivity analysis for the networks edges.

We propose an algorithm that considers every outgoing edge from the Query node. We again use an iterative approach to identify the link strengths that will lead MI's to best matches. For every outgoing edge we increase the value of the strength by θ. The algorithm for computing B and L are repeated with the new weights assigned to the identified edges. The scores (B-L) at each iteration is stored and compared with the score computed with the original weights. The edge which turns out to be most sensitive is the one selected and the recruiter is suggested to strengthen this connection.

Algorithm 4 presents the details of sensitivity analysis. It computes the benefit loss scores on the original network (Step 4). For each link it adds an increment (typically of the order of 0.1) and observes whether the newly computed benefit scores are higher than those obtained on the original graph(Step 5-16). If so, it adds the edge to the list of *BestEdge*(Step 19,20). If set of target nodes is better than the previously obtained network then the link strengths are preserved.

5 EXPERIMENTS AND ASSESSMENTS

Our evaluation concentrates on two important aspects: (i) to show that the extracted MI's are helpful to reach the matching nodes under different settings, viz. identifying MI, MSI and MTI and (ii) that our algorithms is computationally inexpensive compared to a brute force approach. We have used our algorithm on large corpus of Wikipedia data. We proceed by converting the raw data into graphs in the form of adjacency matrix. We have also implemented the algorithm for a talent search application of an organisation that uses employees data for building the network and job description as a query node.

5.1 Information Search -Wikipedia Data

We consider the network provided by the Simple wikipedia first and then on the full large scale Wikipedia Dataset. The nodes refers to the information pages and the edges are the URLs present in the pages. Simple Wikipedia data set had about 147102 documents (nodes) and 1501950 links (Edges) whereas full Wikipedia data set had about 9872158 documents (nodes) and 197443160 links (Edges). Given this network we seek to find the following answers: Given a query string and the page where we currently are rooted (Query node), what are the pages (MI's) we should be directed so that we can reach the matching pages containing the query string. We do not want to reach the non-matching pages for the given query string. We leverage Lucene and its indexing capabilities[1] to compute the matching scores. Each document was extracted into a separate file using the Lucene Wikipedia Extractor function and indexed using the Lucene functions. With the indexes created any desired query string can be searched. Lucene provides a matching score associated with each of them. These scores are considered as matching scores and are normalized to obtain relative measures. We assume uniform probability of one, for all edges because there is no notion of social closeness in this kind of a network. However, this can be adopted to networks that exhibit social closeness and the edge weights can determine the closeness between two nodes. We have implemented a graph building algorithm from raw wikipedia data to be used with our algorithms.

5.1.1 Results and Observation for Identifying Top K Marginal Influencers. We first provide a set of sample queries and report the results obtained for the query. Tables 1(a), (b), 2(a), (b) show the results obtained in terms of three parameters Top M (Matching nodes), Top Influencers and Top Marginal Influencers. Tables 1(a) and 1(b) are queries run on Simple Wikipedia while tables 2(a), (b) were run on full Wikipedia. It can be clearly seen that the Top influences (based on their benefit scores) are completely different than those Top Marginal Influencers (based on their $B - L$ scores). Our algorithm takes usually 4-5 iterations for convergence irrespective of the query and hence, is fast in its computation. Top M nodes simply present the pages of wikipedia that match with the query the best. Upon manual examination of the results, we find that by simply using Top B nodes, the influencers usually turn out to be long lists of elements (or pages with many links), some of which are relevant to our query. Although the Top B nodes may seem more intuitive since they usually are directly related to the query, they aren't as good for our objective since we want to get marginally influential nodes. On the other hand, the Top B-L nodes are usually short pages containing only a few links but most of which are relevant to our query. This shows that, by using our technique, we can simultaneously minimize spam whilst maximizing targeted influence. Table 3 shows the Top B nodes and B-L Nodes when the matching scores are ignored. Our manual inspection shows that in the absence of matching scores, the algorithm behaves like

Table 1: Results on Simple Wiki

(a) "basketball association"

Top M nodes	Top B nodes	Top (B-L) nodes
Denver Nuggets	NBA Development League	Al Albert
Sacramento Kings	NBA All-Star Game	Central Division (NBA)
Indiana Pacers	New York Knicks season	Southwest Division (NBA)
Bob Lamey	NBA Finals	Pacific Division (NBA)
San Antonio Spurs	List of NBA champions	Atlantic Division (NBA)

(b) "jazz music"

Top M nodes	Top B nodes	Top (B-L) nodes
Jazz blues	Mihailo Živanovi	Moaning Lisa
Deep house	Ornette Coleman	Jazz blues
Riff	Billie Holiday	Beat (music)
Sooner or Later	Deaths in 2012	Polyrhythm
Feeling Good	Bandleader	Quartet

Table 2: Results on Full Wiki

(a) "Tennis"

Top M nodes	Top B nodes	Top (B-L) nodes
Tennis (disambiguation)	List of tennis tournaments	2005 World Table Tennis Championships – Women's Singles
List of The Prince of Tennis video games	2009 ATP Challenger Tour	Horatio Pintea
Florida Gators tennis	List of Grand Slam related tennis records	2009 LA Tennis Open USTA Men's Challenger
National Tennis Centre	2008 ATP Challenger Series	Ainslie Tennis Club
Tennis Masters	2011 ATP Challenger Tour	Kjell Johansson

(b) "jazz music"

Top M nodes	Top B nodes	Top (B-L) nodes
Jazz mag	List of NPR stations	Youth Oriented
New Orleans Jazz	List of non-profit radio stations	I Visionari
Jazz piano (disambiguation)	Jazz	Dig it
New Jazz	Outline of jazz	Dig It (disambiguation)
Rebop	Van Gelder Studio	Song of the Heart

Table 3: No Matching Score on Full Wikipedia -jazz music

Top M nodes	Top B nodes	Top (B-L) nodes
Jazz mag	Jazz	Youth Oriented
New Orleans Jazz	List of NPR stations	Hank Bagby
Jazz piano (disambiguation)	List of non-profit radio stations	Tanglewood Stratocaster
New Jazz	Outline of jazz	Tommy Dorsey's Dance Party
Rebop	List of train songs	Blues to Africa

Table 4: Results to Illustrate Top-K *MSI* and Top-K *MTI*

(a) "power rangers"

Top B-L	Strength Max.	Reach Max.
Power Rangers Lightspeed Rescue	Power Rangers Lightspeed Rescue	Power Rangers Lightspeed Rescue
Power Rangers Lost Galaxy	Power Rangers In Space	Time Force
Time Force	Power Rangers Zeo	
Power Rangers In Space	Time Force	
Power Rangers Zeo	Power Rangers Lost Galaxy	

(b) "skype"

Top B-L	Strength Max.	Reach Max.
Iris	Iris	Iris
MSX	Zephyrus	Common Language Specification
Zephyrus	MSX	Microsoft TechNet
Microsoft Paint	Microsoft TechNet	
Professional Developers Conference	Microsoft Paint	

un-labeled networks and therefore does not provide meaningful results.

5.1.2 Results and Observation for Identifying Top K MSI and Top K MTI. On the Wikipedia data, we also derived the rank ordered set of influential vertices. Tables 4 (a) and (b) show sample outputs for two queries. As expected, strength maximization algorithm leads to a selection of nodes closer to the same set of matching

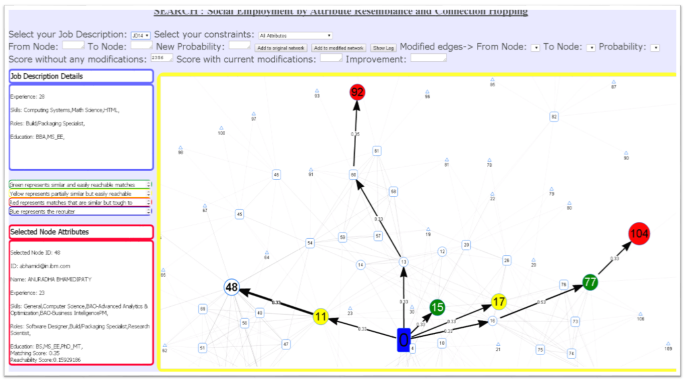

Figure 2: Target Influencers for JD on the employee network.

nodes. This helps in reaching nodes through better strength of connections. This is why there is not much difference between the ranked nodes (MI) and the influential nodes selected MSI. The ranked MI nodes are calculated at once, thus one node may help increase the influence of the other one. But in MSI, we remove the selected nodes one-by-one thereby reducing the influence of the already existing top nodes. This allows the nodes lower in the ranking to come up on top, providing a different set of top ranking MSI. Hence, this algorithm is effective when we want to maximize the probability of reaching the top matching nodes. For MTI, the results show a different phenomenon. In this case, at each iteration, the set of nodes that have already been influenced are removed. The algorithm finds new set of matching nodes in the subsequent iterations. This may lead to MTIs which are lower in the list of MIs. However, they maximize the reach of influence to many non-overlapping matching nodes. When all the matching nodes form a strongly connected subgraph, we expect only one MTI since one influential node would be able to reach all the matching nodes.

5.2 Implementation on Recruitment Network

We have implemented the algorithm for Talent recruitment for an Organization. We have used social network API's of the organization (social networking API (SAND) [5]) to build the network. The nodes have their profile information. Network size ranges from 700 – 10000 in number depending on the level set for discovering network. Given a 'Job Description (JD)' the aim is to identify influencers. As a sample, we have shown in Figure 2 a screen shot of the recruitment tool.

6 CONCLUSION

We have, in this work proposed a practical solution for otherwise imperative and difficult problem of identifying the 'meaningful' influencers in large scale networks. We have considered the scales of the graphs - given that it is applicable to social networks and several application namely recruitment, expertise locator etc., are build on top social networks. Our experiments on the full Wikipedia (around several million nodes) corpus shows that the number of iterations to converge are just 4 - 5 iterations. This makes our algorithm feasible to be used in large graphs in near real time scenarios. Our assessments through experts on the results generated by our algorithm for select queries indicates the superiority of the algorithm proposed.

REFERENCES

[1] Apache. [n. d.]. Lucene Indexer. https://lucene.apache.org/core/.

[2] Marcel Blattner, Yi-Cheng Zhang, and Sergei Maslov. 2007. Exploring an opinion network for taste prediction: An empirical study. *Physica A: Statistical Mechanics and its Applications* 373 (2007), 753–758.

[3] Ralf Caers and Vanessa Castelyns. 2010. LinkedIn and Facebook in Belgium: The influences and biases of social network sites in recruitment and selection procedures. *Social Science Computer Review* (2010), 0894439310386567.

[4] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 199–208.

[5] IBM. [n. d.]. SaND (Social Networks and Discovery). https://www-10.lotus.com/ldd/lcwiki.nsf/xpAPIViewer.xsp?lookupName=IBM+Connections+5.0+API+Documentation&action=openDocument&res_title=Social_Network_(Graph)_API_ic50&content=apicontent.

[6] Zhaoyan Jin, Quanyuan Wu, Dianxi Shi, and Huining Yan. 2013. Random Walk Based Inverse Influence Research in Online Social Networks. In *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*. IEEE, 2206–2213.

[7] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 137–146.

[8] Fa-Hsien Li, Cheng-Te Li, and Man-Kwan Shan. 2011. Labeled influence maximization in social networks for target marketing. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*. IEEE, 560–563.

[9] Yanhua Li, Wei Chen, Yajun Wang, and Zhi-Li Zhang. 2013. Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 657–666.

[10] Kok Lim, Hur Ewing-Chow, Teddy Jasin, Zaidah Mohd, Torres Oey, Wee Ong, Heng Shi, Julian Vincent, Kian Vu, Heling Zhang, et al. 2005. Social marketing network. US Patent App. 11/233,855.

[11] Silviu Maniu and Bogdan Cautis. 2013. Network-aware search in social tagging applications: Instance optimality versus efficiency. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 939–948.

[12] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. (1999).

[13] Rama Kumar Pasumarthi, Ramasuri Narayanam, and Balaraman Ravindran. 2014. Targeted Influence Maximization through a Social Network. (2014).

[14] Ralf Schenkel, Tom Crecelius, Mouna Kacimi, Sebastian Michel, Thomas Neumann, Josiane X Parreira, and Gerhard Weikum. 2008. Efficient top-k querying over social-tagging networks. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 523–530.

[15] Antti Ukkonen, Carlos Castillo, Debora Donato, and Aristides Gionis. 2008. Searching the wikipedia with contextual information. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 1351–1352.

[16] Peifeng Yin, Wang-Chien Lee, and Ken CK Lee. 2010. On top-k social web search. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 1313–1316.

[17] Justin Zhan, Sweta Gurung, and Sai Phani Krishna Parsa. 2017. dentification of top-K nodes in large networks using Katz centrality. *Journal of Big Data* 4, 1, 16. https://doi.org/10.1186/s40537-017-0076-5

[18] Huiyuan Zhang, Thang N Dinh, and My T Thai. 2013. Maximizing the Spread of Positive Influence in Online Social Networks. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*. IEEE, 317–326.