Managing Big Data Using Fuzzy Sets by Directed Graph Node Similarity

Marko Jocic[*], Endre Pap[,***], Anikó Szakál[**], Djordje Obradovic[*], Zora Konjovic[***]**

[*]Faculty of Technical Sciences/Department of Computing and Automation, Trg DositejaObradovića 6, 21000 Novi Sad, Serbia

[**]Óbuda University, Bécsi út 96/b, H-1034 Budapest, Hungary

[***]Singidunum University, Danielova 29, 11000 Belgrade, Serbia

m.jocic@uns.ac.rs, epap@singidunum.ac.rs,.aniko@uni-obuda.hu, obrad@uns.ac.rs, zkonjovic@singidunum.ac.rs

Abstract: This paper proposes a novel algorithm for discovering similar nodes in very large directed graphs, with millions of nodes with billions of connections, which is based on the fuzzy set theory. The required input is a sample of representative nodes that are highly affiliated with some feature. This approach is practically verified on Twitter social network case study to discover influential Twitter users in the field of science.

Keywords: big data; directed graph; node similarity; Twitter

1 Introduction

The explosive growth of social media and massive participation in social networks is reflected in the countless number of contributions that are constantly posted and discussed on social sites such as Facebook, Twitter, Instagram, Pinterest and others. One aspect of special interest is the popularity and status of some members of these social networks measured by the level of attention they receive in terms of followers [1]. The other aspect is the influence that these individual's wield, which is determined by the propagation of their content through the network [2].

Despite an abundance of solutions providing influence measurement, there is still a need for improvements that cope well with the vagueness and uncertainty inherent to data describing influence, while keeping computational efficiency and accuracy of the output results.

In this paper we propose a novel algorithm aimed at discovering similar nodes, in very large directed graphs, that is both efficient and accurate with respect to real-world problems. The algorithm is based on the fuzzy set theory, and its practical application could be, among others, finding influential people in a network.

The paper is organized in the following way. Following this introductory section, the second section presents related works, while the third describes the proposed algorithm. Fourth section brings the Twitter social network case study discovering influential Twitter users in the field of science by which the algorithm is verified. The fifth section gives conclusions and directions for further research on this subject.

2 Related Works

This section presents some popular tools for determining the influence of social media users, and scientific papers using graphs as a model for social networks as well as for determining social network user similarities.

The tool *Socialbakers* [3] tracks, analyses, and benchmarks over 8 million social profiles across all the major social platforms including Facebook, Twitter, YouTube, LinkedIn, Instagram, Google+ and VK. They have statistics that are free of charge and available to everyone with daily updates and historical data up to 3 months. *Simply Measured* [4] is the leading social media analytics platform, providing complete measurement and reporting for serious marketers in all major social platforms including Facebook, Twitter, Google+, Instagram, YouTube, Vine, LinkedIn, and Tumblr. This service delivers profile analytics and audience insights including influence and sentiment analysis. It is available only in paid version, and only samples of data are shown for free. *Trackur* [5], which is also a paid service, allows full monitoring of all mainstream social media including Twitter, Facebook and Google+, but also news, blogs, reviews and forums. This service delivers executive insights including influence scoring. All results are delivered almost in real-time, as many sources are updated every 30 minutes. *Klout* [6] uses Twitter, Facebook, LinkedIn, Wikipedia, Instagram, Bing, Google+, Tumblr, Foursquare, YouTube, Blogger, WordPress, Last.fm, Yammer and Flickr data to create Klout user profiles that are assigned a "Klout Score" ranging from 1 to 100. Higher score corresponds to a higher ranking of the breadth and strength of one's online social influence. Klout is free for influencers, but paid for business users. What is common to all these services is that they are mostly paid, and only just outline algorithms and methods used to infer influence of a certain individual.

Graphs have been used a lot in order to describe social networks and analyze them. A myriad of techniques and approaches to social network analysis, data mining, graph mining and representation of social networks as graph structures

can be found in [7]. There are also papers about determining important network characteristics [8], [9] and graph properties [10] that have a potential for practical implementations in various domains including social networks. Similarity measures between objects are of central importance for various data analysis techniques including the special case of similarity measures related to graphs. A number of measures for such purpose have been proposed, especially for calculation of similarity of graphs nodes [11], [12], [13], [14], [15], [16], [20]. These methods have been successfully applied in several domains like ranking of query results [11], synonym extraction [13], database structure matching [17], construction of phylogenetic trees [12], analysis of social networks [18], [19], recommending trustworthy agents in a trust network [20], etc. There is a vast number of researches that try to determine the measure of influence in various social media and social networks [2], [21], [22]. This leads to many measures of influences, where some of these are correlated and some are not.

What is important for realistic modeling of influence in social media/networks is to capture the temporal dynamics (new influential users appearing over time, some users losing their influence over time), and the fact that attributes determining the measure of influence may be subjective, with values which are often imprecise or even vague. These imprecise data, in the context of social networks, and more specifically relations between users, their interests and influence can be modeled by fuzzy graphs [23], [24].

This encouraged the authors of this paper to adopt a fuzzy-like approach to the problem in order to model imprecise knowledge about influence within a social network.

3 Construction of the Algorithm

In this section we present the proposed algorithm (the basic one, and the one with reduced computational complexity) preceeded by preliminaries aimed mainly at introducing basic notions, and the notation that is used throughout the paper.

3.1 Preliminaries

We denote a directed graph by G, i.e. ordered pair $G = (N, E)$, where N is a set of all nodes in the graph, and E is a set of all edges in the graph. Edge $e = (a, b)$ is an ordered pair of two nodes and is directed from node a to node b, where a is the source node, and b is the terminating node. Another notion is that b is a direct successor of a, and a is said to be a direct predecessor of b. We denote by $|N|$ the cardinality of a graph (the total number of nodes), and by $|E|$ a graph size (the total number of edges/ connections between nodes). For each node in a directed

graph, there is an *in-degree* – the number of edges coming "into" certain node, and *out-degree* – the number of edges coming "out" of certain node. A *degree* is the sum of in-degree and out-degree. Individual elements of all sets will be denoted with lowercase letters corresponding to the name of the set, along with subscript index, e.g., set N has elements $n_i, i \in [1, |N|],\ i \in \mathbb{N}$, where i represents an index of i-th element in set N. All direct predecessors of some node a are noted as the set DP(a). This means that the in-degree of node a can be represented as the cardinality of DP(a). Obviously, DP(a) ⊂ N. Also, individual elements of DP(a) will be denoted by $dp_i(a)$, i.e.,

$$DP(a) = \{z | (z, a) \in E \ \wedge \ z, a \ \in N\} \tag{1}$$

$$indegree(a) = |DP(a)| \tag{2}$$

All direct successors of some node a are noted as a set $DS(a)$. This means that the out-degree of node a can be represented as a cardinality of $DS(a)$. Also, $DS(a) \subset N$. Individual elements of $DS(a)$ will be denoted $ds_i(a)$, i.e.,

$$DS(a) = \{z | (a, z) \in E \ \wedge \ z, a \ \in N\} \tag{3}$$

$$outdegree(a) = |DS(a)| \tag{4}$$

Following the fuzzy set theory [25], there is a fuzzy set described by its membership function μ_R, which here corresponds to the affiliation with some feature. Every node a in graph G has a value in this membership function $\mu_R(a)$,

$$\mu_R(a) \in [0,1] \wedge \mu_R(a) \in \mathbb{R}, a \in N \tag{5}$$

Here, a value of 0 means that no affiliation with certain feature exists, a value of 1 means utter affiliation with certain feature, and the values between 0 and 1 render partial association with a certain feature - the higher the value, the greater the affiliation. It can also be said that two nodes with similar values of the membership function for a particular feature, also exhibit some similarity between each other regarding that feature.

It is important to note that this directed graph *G* is large in a sense that it has a huge number of nodes and an even larger number of directed connections, where the number of nodes is at least a million and the number of connections is on the order of several billion. This introduces difficulties from the computational resources point of view. The proposed algorithm is designed to work with a (parts of) large directed graphs, where average $indegree$ of nodes exceeds the average $outdegree$ of nodes by several orders of magnitude.

3.2 Basic Algorithm

The algorithm starts by selecting representative nodes in a graph that have high association with some desired feature. These nodes should be selected by expert or experts in the particular domain in which this feature is exposed. This set of

representative nodes will be denoted as $R, R \subset N$, where each node in R will be denoted $r_i, i \in [1, |R|]$. Based on expert's knowledge, every selected node r_i is assigned its membership function value $\mu_R(r_i)$, which depicts its affiliation with the desired feature. Now, the procedure should be able to find nodes in a graph that are similar to the selected nodes – similar in their membership function values μ_R. The creation of this representative set and assigning membership function values can be regarded more formally as obtaining of a training set for a supervised machine learning algorithm. After selecting representative nodes, for each of these nodes, the algorithm obtains all its direct predecessors. The result is a union (set) of all direct predecessors for all representative nodes; this set will be denoted as DPR, and its elements will be denoted as dpr_i, i.e.,

$$DPR = \bigcup_{i=1}^{|R|} DP(r_i) \tag{6}$$

Many of the representative nodes have some common direct predecessors, which imply that many of these predecessors in DPR will have common direct successors among the representative nodes in R. As these direct successors in R are subset of all direct successors, a new set is introduced, $DSR\ (a) = DS(a) \cap R$. This intersection $DSR\ (a)$ results in only those direct successors of certain node a that are in representative set R, too. Now, for each of these nodes in a set DPR a value v is calculated by summing membership function values of nodes in R which are direct successors to a certain node:

$$\forall dpr_i \in DPR \Rightarrow\ v(dpr_i) = \sum_{dsr \in DSR(dpr_i)} \mu_R\ (dsr) \tag{7}$$

The calculated values are refined by taking in an account the out-degree for each node in DPR. More precisely, the calculated value v of a node in DPR is divided with its out-degree, resulting in a new value vr:

$$vr(dpr_i) = \frac{v(dpr_i)}{|DS(dpr_i)|} \tag{8}$$

When all nodes in DPR have had their values vr calculated, the algorithm can proceed to the next step. For each of the nodes in DPR, all its direct successors are obtained, i.e. the nodes that it is directly connected to. The result of this is a union (set) of all direct successors for all nodes in DPR. This set will be denoted as DSU, and its elements will be denoted as dsu_i, i.e.

$$DSU = \bigcup_{i=1}^{|DPR|} DS(dpr_i) \tag{9}$$

It can be noted here that DSU is a superset of R, $DSU \supset R$, because it will certainly contain all the representative nodes, but also other nodes that weren't marked as representative by the expert. After obtaining DSU, for each node in DSU, a value s is calculated by summing all of the already calculated values vr of its direct predecessors in DPR. This summed value can be explained as a similarity measure between certain node in DSU and nodes in R, i.e.,

$$\forall dsu_i \in DSU \Rightarrow\ s(dsu_i) = \sum_{dp \in DP(dsu_i)} vr\ (dp) \tag{10}$$

One final step must be taken in order to complete the calculation. If one takes two arbitrary nodes from DSU, where both of these have the same calculated value, but have different in-degree (number of all direct predecessors), how can that be interpreted? The node with smaller in-degree should expose more similarity with representative nodes in R than the node with larger in-degree. The explanation lies in absolute representation of this similarity, as it is just merely a sum, while some relative representation might contain more information. Thus, each of the calculated values is recalculated by dividing it with the node's in-degree, resulting in value sr, i.e.,

$$sr(dsu_i) = \frac{s(dsu_i)}{|DP(dsu_i)|} \tag{11}$$

First, the value sr is calculated for all of the representative nodes, in order to examine if calculated similarity values correspond to the originally assigned values by the expert. Here, it was noticed that the calculated values are actually smaller than the assigned values, which is a consequence of representative set being smaller than average in-degree of nodes in the representative set. In some cases, the calculated values are just two times smaller, but in larger graphs with lots of connections, this ratio might be a much larger value. In order to correct the calculation, new value is introduced - a correction coefficient c. This coefficient is actually a mean ratio between the assigned value and the calculated value, i.e.,

$$c = \frac{\sum_{r \in R} \mu_R(r)/sr(r)}{|R|} \tag{12}$$

The final measure of similarity denoted by $srf(a)$, or the value of membership function is then calculated by multiplying the calculated value sr with the correction coefficient c, i.e., $srf(a) = c * sr(a)$. The resulting value srf is basically membership function value for each node in DSU, where nodes that have the smallest values of this number shouldn't manifest similarity with the feature exposed among nodes in R, and vice versa. The calculated similarity measure is actually a value of membership function that represents affiliation with the nodes in the representative set $\mu_R(a) = srf(a)$. Now the expert can select the nodes from DSU with the highest calculated similarity and examine if the desired feature is present, as it was in representative set R.

The above described algorithm is more concisely represented by Pseudo-code 1.

Pseudo-code 1:
Basic algorithm

```
choose set of representative nodes
foreach node r in R:
  assign membership function value for r

init DPR = empty set
```

```
foreach node r in R:                    (equation 6)
  DP = fetch_direct_predecessors(r)
  DPR = find_union(DP, DPR)

foreach node dpr in DPR:        (equations 7 and 8)
  DS = fetch_direct_successors(dpr)
DSR = intersection(DS, R)
  init v = 0
  foreach node dsr in DSR:
    v = v + membership(dsr, R)
  vr(dpr) = v / size(DS)
  init DSU = empty set
foreach node dpr in DPR:                (equation 9)
  DS = fetch_direct_successors(dpr)
  DSU = find_union(DS, DSU)
```

Pseudo-code 1:
Basic algorithm (continuation)

```
foreach node dsu in DSU:     (equations 10 and 11)
  DP = fetch_direct_predecessors(r)
  init s = 0
  foreach node dp in DP:
    s = s + vr(dp)
  sr(dsu) = s / size(DP)

init c = 0
foreach node r in R:                  (equation 12)
  c = c + sr(r)
c = c / size(R)

foreach node dsu in DSU:
  srf(dsu) = c * sr(dsu)
```

3.3 Algorithm with Reduced Computational Complexity

Computational complexity is an important issue of any implementation that intends to reach performance satisfying real-life demands. The algorithm that is described in previous subsection is not exception to this. This means that performance indicators on quality characteristics like precision should be balanced with performance indicators like resource's consumptions.

There are few things that one needs to consider while determining computational complexity of the proposed algorithm.

- First, the running time of a union operation in equation (6) depends on sum of in-degree values for each node in representative set, thus the running time of this part is $O(\sum_{r \in R} indegree(r)) = O(|DPR|)$.
- Second, running time of calculating values $v(dpr_i), vr(dpr_i)$ from equations (7) and (8), respectively, depends on a number of nodes in DPR set, $n_{DPR} = |DPR|$, and number of nodes in DSR set, $n_{DSR} = |DSR|$, which results in running time for this part of $O(n_{DPR} * n_{DSR})$. But since $DSR \subset R \Rightarrow |DSR| < |R|$, and $|R| \ll |DSR|$, running time can be approximated as $O(n_{DPR}) = O(|DPR|)$
- Third, the running time of the union operation in equation (9) depends on the sum of out-degree values for each node in DPR set, which makes running time of this part $O(\sum_{dpr \in DPR} outdegree(dpr))$. Again, since this algorithm works with parts of graphs which usually have in-degree values with at least two orders of magnitude larger than out-degree values, running time for this part can be approximated as $O(|DPR|)$ as well.
- Fourth, running time of calculating values $s(dsu_i), sr(dsu_i)$ from equations (10) and (11), respectively, depends on total number of nodes in DSU set, and total number of direct predecessors for each node in DSU set. But, because this part of the algorithm iterates through direct predecessors that are already in the DPR set, one can say that running time of this part is also O(|DPR|).

Finally, by summing all these running times, we get $O(4 * |DPR|) = O(|DPR|)$, which leads to the conclusion that computational complexity of the entire algorithm depends mostly on the number of taken direct predecessors of nodes in the representative set.

Since the defined (part of) large directed graph *G* typically has nodes with high in-degree values, this makes the whole approach computationally expensive. This indicates that, in order to reduce the running time, one could try to take smaller portion of these direct predecessors, i.e. set $SDPR \subset DPR$, which contains randomly sampled elements of DPR. Here the cardinality of $SDPR$ is defined as $|SDPR| = \rho * |DPR|, \rho \in [0,1]$, and parameter ρ denotes the portion of randomly sampled elements.

By adopting this approach in the paper we created an algorithm modification, which is presented by the Pseudo-code 2 where the bolded content is for modifications.

Pseudo-code 2:
Algorithm modification for reduction of computational complexity

```
choose set of representative nodes

foreach node r in R:
  assign membership function value for r
```

```
init DPR = empty set
foreach node r in R:
   DP = fetch_direct_predecessors(r)
   DPR = find_union(DP, DPR)

SDPR = random_sample(DPR, rho)

foreach node dpr in SDPR:
DS = fetch_direct_successors(dpr)
   DSR = intersection(DS, R)
   init v = 0
   foreach node dsr in DSR:
      v = v + membership(dsr, R)
   vr(dpr) = v / size(DS)
init DSU = empty set
```

Pseudo-code 2:
Algorithm modification for reduction of computational complexity (continuation)

```
foreach node dpr in SDPR:
   DS = fetch_direct_successors(dpr)
   DSU = find_union(DS, DSU)

foreach node dsu in DSU:
   DP = fetch_direct_predecessors(r)
   init s = 0
   foreach node dp in DP:
      s = s + vr(dp)
   sr(dsu) = s / size(DP)

init c = 0
foreach node r in R:
   c = c + sr(r)
c = c / size(R)

foreach node dsu in DSU:
   srf(dsu) = c * sr(dsu)
```

4 Twitter Case Study

In this section we present a case study aimed at demonstrating the algorithm, and to show that the proposed modification of the basic algorithm performs better in terms of resources consumption, retaining, at the same time, acceptable precision.

Since the proposed algorithm was verified on the Twitter social network, in the first subsection of this section some notions are given about this social network and its data. The second subsection presents application of our algorithm to Twitter.

4.1 Twitter and Its Data

Twitter is one of the biggest social networks with more than 300 million active users per month, which makes it second largest social network, after Facebook. Twitter users follow others or are followed. Unlike on most online social networking sites, such as Facebook or MySpace, the relationship of following and being followed requires no reciprocation. A user can follow any other user, and the user being followed doesn't need to follow back. Being a follower on Twitter means that the user receives all the messages (called tweets) from those the user follows. Common practice of responding to a tweet has evolved into well-defined markup culture: RT stands for retweet, '@' followed by a user identifier addresses the user, and '#' followed by a word represents a hashtag. This well-defined markup vocabulary combined with a strict limit of 140 characters per posting conveniences users with brevity in expression. The retweet mechanism empowers users to spread information of their choice beyond the reach of the original tweet's followers [26].

Twitter, and social networks in general, can easily be represented as a graph, where Twitter is an example of a directed graph.

Twitter REST API provides programmatic access to read and write Twitter data [27]. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON. In order to perform the proposed algorithm, one should be able to obtain: profiles of certain Twitter users (profile contains basic information, as well as the number of followers and number of following), followers of certain user, and users that certain user is following. All of these data are obtained via Twitter REST API. The base URI for all Twitter REST API calls is https://api.twitter.com/1.1.

It is also important to mention that the calls to REST API are rate limited – all methods allow only a limited number of calls within a 15 minute window. Rate limiting is primarily considered on a per-user basis, or more accurately, per access token in your control. If a method allows for 15 requests per rate limit window, then it allows you to make 15 requests per window per leveraged access token [28]. This rate limiting raises practical difficulties when trying to download big

amounts of data from Twitter because it simply takes a lot of time – one should carefully orchestrate calls to the REST API within these 15 minutes windows in order to make the most use of Twitter REST API and its data.

For fetching Twitter users' profiles, HTTP request GET https://api.twitter.com/1.1/users/lookup is called, which returns fully-hydrated user objects for up to 100 users per request, as specified by comma-separated values passed to the user_id and/or screen_name parameters. Rate limit for this endpoint is 180 calls per 15 minute window.

In order to obtain followers of a certain Twitter user, an HTTP request GET https://api.twitter.com/1.1/followers/ids is called, which returns a cursored collection of user IDs for every user following the specified user. Results are given in groups of 5,000 user IDs and multiple "pages" of results can be navigated through using the next_cursor value in subsequent requests. Rate limit for this endpoint is 15 calls per 15 minute window.

4.2 Discovering Similar Influential Users on Twitter

The proposed algorithm was used to discover similar influential users in Twitter social network.

The process starts with selecting a group of representative users. In this experiment a group of 163 Twitter users was selected, based on their influence in the category of science. Influential science users engage their followers with news and interactive tweeting in many spheres of science. For each of these users, an expert assigns a membership function value, which determines how much certain user belongs to this representative group.

Some users of this representative group of Twitter users influential in science domain are shown in the Table 1.

Table 1
A sample of influential Twitter users in science

Name	**Twitter screen name**	**Membership function value**
NASA	@nasa	0.9
Scientific American	@sciam	0.85
New Scientist	@newscientist	0.85
WIRED Science	@wiredscience	0.7
Neil deGrasse Tyson	@neiltyson	0.8
CERN	@cern	0.9
National Geographic	@natgeo	0.7
Curiosity Rover	@marscuriosity	0.8
Phil Plait	@badastronomer	0.85
Richard Dawkins	@richarddawkins	0.7

After selecting a group of representative users, all their followers must be downloaded from the Twitter REST API. This is the most resource (time and storage) consuming part of the whole process, as most of these users have more than 1 million followers, while some of them have even more than 10 million followers. This implies that for most users, in order to get all of their followers, the Twitter REST API must be called at least 200 times (200 x 5K followers = 1M followers). Bearing in mind that for fetching followers only 15 requests per 15 minute window is allowed by the Twitter REST API, the estimated needed time to download 1 million followers with one Twitter access token is 3 hours and 20 minutes. Also, each request returns 5,000 IDs of followers, where ID is a 64-bit number. This results in need of at least 8B x 1M followers = 8,000,000 B ~ 7,5MB of storage for single user's followers, and that is only for users with 1 million followers.

Note that needed resources (time and storage) increase linearly with number of users' followers. However, this still imposes technical and practical difficulties on huge graphs with a large number of connections, like the Twitter social network. It is also important to note that lots of these followers are not unique for certain representative user. It is very likely that two Twitter users that promote science and that have more than 1 million followers will have some (or many) common followers. This implies that many of these followers will follow more than just one user from the representative group.

After downloading all the representative users' followers, for each of these followers a relative value is calculated that describes his/her "interest" in this representative group of users, in this case specifically – in science. The way this is done is trying to utilize relevant data (the number of users that promote science followed by a certain user as well as representative users' membership functions), and, at the same time, neutralize what's called *aggressive following*[1] on Twitter.

In order to calculate this value, the total number of friends (users that a certain user follows) has to be downloaded. This is done by obtaining the whole Twitter profile via Twitter REST API for all followers, but storing just total number of friends. After obtaining the total number of friends, a value that should give more insight about certain user interests is calculated.

Following the calculation of followers' interest in the representative group, by examining the data it can be seen that many of the users with the highest calculated value of interest really are engaged a lot in the Twitter scientific community, which means they often tweet, retweet and favorite scientific articles, facts, news, etc.

When all followers are downloaded and their values of interest are calculated, the process proceeds with downloading all of the followers' friends – users that

[1] *Aggressive following* is defined as indiscriminately following hundreds of accounts just to garner attention.

followers follow. Unlike downloading followers, downloading friends is much less resource consuming because according to Twitter, every account can follow 2,000 users in total. Once someone has followed 2,000 users, there are limits to the number of additional users can follow. This number is different for each account and is based on user's ratio of followers to following; this ratio is not published. Follow limits cannot be lifted by Twitter and everyone is subject to these limits, even high profile and API accounts. That said, just one request to the Twitter REST API should suffice in order to obtain certain user's friends.

When the process of downloading followers' friends is complete, for each of these friends a value is calculated by summing all values of interest already calculated for their followers. This new value could be interpreted as how much certain user's followers are interested in science, or more generally whatever feature the representative group is affiliated to. We call this value *absolute similarity*. Dividing this absolute similarity by the number of followers results in a kind of "per follower influence", which we call *relative similarity*.

Now, for each of the representative users, the calculated value of similarity can be compared to the originally assigned value of membership function. What occurred in our experiment was that the calculated values were much smaller than the assigned values, with mean ratio between the assigned and the calculated value $c = 2.57$. Therefore, the final similarity value is then calculated by multiplying the previously calculated relative similarity by the correction coefficient *c*. This final similarity is in regards to Twitter users in the representative group, or more specifically influential Twitter users in the field of science.

Here the algorithm stops and the expert inspects the calculated data most probably by looking at the users with the highest calculated value of influence.

Of course, Twitter users that were not in the representative group are especially interesting for examination because they are new influential Twitter users not known as such previously. In addition to discovering new users, the results might show that certain Twitter users that were originally put in the representative group are actually not that influential at all. This gives expert an opportunity to review and revise the data (for example, newly discovered influential users can be added to the representative group) and the algorithm can be re-run hopefully yielding even more discovery.

The results of discovering similar influential Twitter users in science show that the proposed algorithm indeed works as intended.

When provided with 163 representative influential Twitter users in the category of science, a total of 72 new users were discovered with a similarity measure above 0.5.

Some discovered users are shown in Table 2.

Table 2
A sample of discovered influential Twitter users in science

Name	Twitter screen name	Description	Similarity
Robert Simpson	@orbitingfrog	Astronomer	0.84
Stuart Clark	@drstuclark	Astronomer, journalist	0.71
CaSE	@sciencecampaign	The campaign for Science & Engineering	0.73
Chemistry World	@chemistryworld	Chemistry magazine	0.6
Armed with Science	@armedwscience	US Defense Depart. science and technology blog	0.68
NASA's Juno Mission	@nasajuno	NASA's mission to Jupiter	0.78
ESA Science	@esascience	European Space Agency science blog	0.81

Table 3 shows a comparison of the calculated values of membership function of the representative users with those assigned by the expert. Many of the calculated values don't deviate much from the assigned values. The root error of the mean square error for all 163 representative users is 0.09.

Table 3
A comparison between assigned and calculated value of membership function of sampled representative users

Name	Twitter screen name	Assigned value	Calculated value
NASA	@nasa	0.9	0.82
Scientific American	@sciam	0.85	0.71
New Scientist	@newscientist	0.85	0.82
WIRED Science	@wiredscience	0.7	0.77
Neil deGrasse Tyson	@neiltyson	0.8	0.84
CERN	@cern	0.9	0.79
National Geographic	@natgeo	0.7	0.66
Curiosity Rover	@marscuriosity	0.8	0.69
Phil Plait	@badastronomer	0.85	0.69
Richard Dawkins	@richarddawkins	0.7	0.83

4.3 Performance Improvement

The previous subsection has shown that it is possible to find similar Twitter users by feeding the proposed algorithm with a representative group of Twitter users assuming download of all her/his followers. This section attempts to find out how

many followers per Twitter user are enough to be downloaded, but still be able to get results comparable with those when all followers were downloaded.

Let's say that the entire process of discovering 72 users with a similarity measure above 0.5 from 163 representative users in the previous experiment was performed in some reference time *T*. One could expect that, by taking a smaller portion of all followers, a smaller number of users with a similarity measure bigger than 0.5 will be found. So, let's say that the discovered 72 users with a similarity measure bigger than 0.5 is the highest precision possible with the provided data, because all followers are taken, and this number of 72 users is considered as a reference value for precision, which will be denoted *D*. Hence, the precision when all followers are taken is $p = 1$, when $D = 72$.

In order to test if the algorithm performs well even if fewer followers are taken, it was run again by downloading 50%, 25% and 10% of all followers for each representative Twitter user. This portion of followers taken is denoted as ρ, where $\rho \in [0,1] \wedge \rho \in \mathbb{R}$. However, it is important to note that by taking only a portion of followers, the calculated similarity measure is expected to be proportionally smaller. Therefore the calculated similarity is multiplied by corresponding multiplier, i.e.:

$$\mu_R(dsu_i) = \frac{sr(dsu_i)}{\rho} \tag{13}$$

Table 4 shows the number of discovered users with similarity measure above 0.5, the precision relative to the highest precision possible, the time needed for the whole process, precision to time ratio and the root mean square error (RMSE) for the representative users.

Table 4
Algorithm performance improvement results comparison

Portion of followers taken (ρ)	**Discovered users (D)**	**Precision (p)**	**Time (T)**	**p/T**	**Correction coefficient (c)**	**RMSE**
1	72	1	1	1	2.57	0.09
0.5	67	0.93	0.54	1.72	6.13	0.11
0.25	58	0.8	0.28	2.86	12.33	0.13
0.1	51	0.71	0.12	5.92	27.65	0.18

As expected, even by taking less followers, comparable results are obtained in much less time. By taking 10 times less followers (10%), the algorithm discovered 71% of the users that were discovered by taking all followers, but consuming almost 10 times fewer resources. The root mean square error was only doubled by taking 10% of the followers.

The main advantage of running the algorithm by taking less number of followers is, of course, the less use of resources, primarily time. This allows an expert to run the process by taking just 10% of followers, thus saving almost 90% time, but sacrificing only 30% precision. Also, the discovered users can be added to the representative set and then the algorithm can run again iteratively, discovering even more users.

Conclusions

This paper proposes a novel algorithm for finding similar nodes in directed graphs. The algorithm needs to be provided with the representative set of nodes that expose some feature which should be discovered among other nodes in graph. The representative set of nodes is described by values of their membership function, which corresponds to the affiliation of the node with the desired feature. The calculated measure of similarity with the representative nodes correlates to the true value of the membership function. The algorithm is verified on the Twitter social network case study by discovering influential Twitter users in the field of science. A set of 163 representative influential Twitter users is fed to the algorithm, which results in discovery of new 72 influential users in science.

Also, a proposal is made aimed at improving the algorithm's efficiency in terms of time and storage complexity, which relies upon assumption that many of the members in the representative group share many common followers (which evidently holds for the presented case study). The preliminary results indicate that by reducing the number of downloaded followers to only 10% of the original set the algorithm yielded comparable results with a tolerable increase in error. Bearing in mind that influence in social media is very dynamic and vague concept, the algorithm can be refined by a series of measures: user popularity (number of followers), user activity (number of tweets), followers engagement on user activity (number of retweets, favorites), combination of user popularity and activity, etc. Also, each of these measures change over time, and these changes could be tracked in order to provide some valuable trend information. By taking some of these measures into account to certain Twitter user's influence, it is reasonable to assume that the whole algorithm could yield better results.

References

[1] D. M. Romero, W. Galuba, S. Asur, and B. A. Huberman, "Influence and Passivity in Social Media," in Machine Learning and Knowledge Discovery in Databases, Springer, 2011, pp. 18-33

[2] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi, "Measuring User Influence in Twitter: The Million Follower Fallacy," ICWSM, Vol. 10, pp. 10-17, 2010

[3] "Social Media Marketing, Statistics & Monitoring Tools," Socialbakers.com. [Online]. Available: http://www.socialbakers.com/. [Accessed: 28-Dec-2014]

[4] "Simply Measured | Easy Social Media Measurement & Analytics," Simply Measured. [Online]. Available: http://simplymeasured.com/. [Accessed: 28-Dec-2014]

[5] "Social Media Monitoring Tools & Sentiment Analysis Software," Trackur. [Online]. Available: http://www.trackur.com/. [Accessed: 28-Dec-2014]

[6] "Klout | Be Known For What You Love," Klout. [Online]. Available: https://klout.com/home. [Accessed: 28-Dec-2014]

[7] D. F. Nettleton, "Data mining of social networks represented as graphs," Comput. Sci. Rev., Vol. 7, pp. 1-34, Feb. 2013

[8] A. Rusinowska, R. Berghammer, H. De Swart, M. Grabisch, "Social Networks: Prestige, Centrality, and Influence (Invited paper)". de Swart. RAMICS 2011, Springer, pp.22-39, 2011, Lecture Notes in Computer Science (LNCS) 6663. <hal-00633859>

[9] V. Halasz, L. Hegedus, I. Hornyak, B. Nagy: Solving Application Oriented Graph Theoretical Problems with DNA Computing. In Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), AISC 201 (Springer) 2012, pp. 75-85

[10] T. Réti, I. Felde "On Some Properties of Pseudo-Semiregular Graphs," Acta Polytechnica Hungarica, Vol. 13, No. 6, pp. 45-65, 2016

[11] J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," J. ACM JACM, Vol. 46, No. 5, pp. 604-632, 1999

[12] M. Heymans and A. K. Singh, "Deriving Phylogenetic Trees from the Similarity Analysis of Metabolic Pathways," Bioinforma. Oxf. Engl., Vol. 19 Suppl 1, pp. i138-146, 2003

[13] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren, "A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching," SIAM Rev., Vol. 46, No. 4, pp. 647-666, Jan. 2004

[14] L. A. Zager and G. C. Verghese, "Graph Similarity Scoring and Matching," Appl. Math. Lett., Vol. 21, No. 1, pp. 86-94, Jan. 2008

[15] M. Nikolic, "Measuring Similarity of Graph Nodes by Neighbor Matching," Intell. Data Anal., Vol. 16, No. 6, pp. 865-878, 2012

[16] L. Kovács, G. Szabó, "Conceptualization with Incremental Bron-Kerbosch Algorithm in Big Data Architecture" Acta Polytechnica Hungarica, Vol. 13, No 2. pp. 139-158, 2016

[17] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching," in Data Engineering, 2002. Proceedings. 18th International Conference on, 2002, pp. 117-128

[18] E. A. Leicht, P. Holme, and M. E. Newman, "Vertex Similarity in Networks," Phys. Rev. E, Vol. 73, No. 2, p. 26120, 2006

[19] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Effects of User Similarity in Social Media," in Proceedings of the fifth ACM international conference on Web search and data mining, 2012, pp. 703-712

[20] C.-W. Hang and M. P. Singh, "Trust-based Recommendation Based on Graph Similarity," in Proceedings of the 13th International Workshop on Trust in Agent Societies (TRUST). Toronto, Canada, 2010. [Online]. Available: https://www.csc2.ncsu.edu/faculty/mpsingh/papers/mas/aamas-trust-10-graph.pdf

[21] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring Networks of Diffusion and Influence," ACM Trans. Knowl. Discov. Data, Vol. 5, No. 4, pp. 1-37, Feb. 2012

[22] J. Leskovec, A. Singh, and J. Kleinberg, "Patterns of Influence in a Recommendation Network," in Advances in Knowledge Discovery and Data Mining, Springer, 2006, pp. 380-389

[23] M. S. Sunitha and S. Mathew, "Fuzzy Graph Theory: a Survey," Ann. Pure Appl. Math., Vol. 4, No. 1, pp. 92-110, 2013

[24] J. Vascak, L. Madarasz, "Adaption of Fuzzy Cognitive Maps - a Comparison Study" Acta Polytechnica Hungarica, Vol. 7, No. 3, pp. 109-122, 2010

[25] L. A. Zadeh, "Fuzzy Sets," Inf. Control, Vol. 8, pp. 338-353, 1965

[26] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a Social Network or a News Media?," in Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 591-600

[27] "REST APIs," Twitter Developers. [Online]. Available: https://dev.twitter.com/rest/public. [Accessed: 26-Aug-2015]

[28] "API Rate Limits," Twitter Developers. [Online]. Available: https://dev.twitter.com/rest/public/rate-limiting. [Accessed: 26-Aug-2015]