Big Data Analytics of Social Networks for the Discovery of "Following" Patterns

Carson Kai-Sang Leung(✉) and Fan Jiang

University of Manitoba, Winnipeg, MB, Canada
kleung@cs.umanitoba.ca

Abstract. In the current era of big data, high volumes of valuable data can be easily collected and generated. Social networks are examples of generating sources of these big data. Users (or social entities) in these social networks are often linked by some interdependency such as friendship or "following" relationships. As these big social networks keep growing, there are situations in which individual users or businesses want to find those frequently followed groups of social entities so that they can follow the same groups. In this paper, we present a big data analytics solution that uses the MapReduce model to mine social networks for discovering groups of frequently followed social entities. Evaluation results show the efficiency and practicality of our big data analytics solution in discovering "following" patterns from social networks.

1 Introduction and Related Works

Nowadays, high volumes of valuable data can be easily collected or generated from different sources such as social networks. Social networks are generally made of social entities (e.g., individuals, corporations, collective social units, or organizations) that are linked by some specific types of interdependencies (e.g., kinship, friendship, common interest, beliefs, or financial exchange). A social entity is connected to another entity as his next-of-kin, friend, collaborator, co-author, classmate, co-worker, team member, and/or business partner. Big data analytics of social networks computationally facilitates social studies and human-social dynamics in these big data networks, as well as designs and uses information and communication technologies for dealing with social context.

In the current era of big data (including big social network data), various social networking sites or services—such as Facebook, Google+, LinkedIn, Twitter, and Weibo [16,17]—are commonly in use. For instance, Facebook users can create a personal profile, add other Facebook users as friends, exchange messages, and join common-interest user groups. The number of (mutual) friends may vary from one Facebook user to another. It is not uncommon for a user A to have hundreds or thousands of friends. Note that, although many of the Facebook users are linked to some other Facebook users via their mutual friendship (i.e., if a user A is a friend of another user B, then B is also a friend of A), there are situations in which such a relationship is not mutual. To handle these situations, Facebook added the functionality of "follow", which allows a user

S. Madria and T. Hara (Eds.): DaWaK 2015, LNCS 9263, pp. 123–135, 2015.
DOI: 10.1007/978-3-319-22729-0_10

to subscribe or follow public postings of some other Facebook users without the need of adding them as friends. So, for any user C, if many of his friends followed some individual users or groups of users, then C might also be interested in following the same individual users or groups of users. Furthermore, the "like" button allows users to express their appreciation of content such as status updates, comments, photos, and advertisements. For example, when we liked the page "DEXA Society" (for information about DaWaK), many of our friends might also be interested in this page.

Similarly, Twitter users can read the tweets of other users by "following" them. Relationships between social entities are mostly defined by following (or subscribing) each other. Each user (social entity) can have multiple followers, and follows multiple users at the same time. The follow/subscribe relationship between follower and followee is not the same as the friendship relationship (in which each pair of users usually know each other before they setup the friendship relationship). In contrast, in the follow/subscribe relationship, a user D can follow another user E while E may not know D in person. For instance, a participant attending the DaWaK conference can follow @DEXASociety, but may not be followed by it. This creates a relationship with direction in a social network. We use D→E to represent the follow/subscribe (i.e., "following") relationship that D is following E.

In recent years, the number of users in the aforementioned social networking sites has grown rapidly (e.g., 1.44 billion monthly active Facebook users and 302 million monthly active Twitter users at the end of March 2015). This big number of users creates an even more massive number of "following" relationships. Over the past two decades, several data mining algorithms and techniques [1,5,8–10] have been proposed. Many of them [7,11,15] have been applied to mine social networks (e.g., discovery of special events [3], detection of communities [13,19], subgraph mining [20], as well as discovery of popular friends [6,11], influential friends [12] and strong friends [18]). In DaWaK 2014, we [4] proposed a *serial* algorithm to mine interesting patterns from social networks. While such an algorithm works well when mining a small focused portion of a social network due to its serial nature, there are situations in which one wants to mine a larger portion of a big social network. In response, we propose in the current DaWaK 2015 paper a new big data analytics and mining solution, which uses the MapReduce model to discover *interesting/popular "following" patterns* consisting of social entities (or their social networking pages) that are frequently followed by social entities. Such discovery of "following" patterns helps an individual user find popular groups of social entities so that he can follow the same groups. Moreover, many businesses have used social network media to either (i) reach the right audience and turn them into new customers or (ii) build a closer relationship with existing customers. Hence, discovering those who follow collections of popular social networking pages about a business (i.e., discovering those who care more about the products or services provided by a business) helps the business identify its targeted or preferred customers.

The remainder of this paper is organized as follows. The next section provides some background. Then, we present our new big data analytics and mining solution, which uses the MapReduce model to discover interesting "following" patterns from big social networks in Sect. 3. Evaluation results are shown in Sect. 4. Finally, conclusions are given in Sect. 5.

2 Background

High volumes of valuable data (e.g., web logs, texts, documents, business transactions, banking records, financial charts, medical images, surveillance videos, as well as streams of marketing, telecommunication, biological, life science, and social media data) can be easily collected or generated from different sources, in different formats, and at high velocity in many real-life applications in modern organizations and society. This leads us into the new era of *big data* [14], which refer to high-veracity, high-velocity, high-value, and/or high-variety data with volumes beyond the ability of commonly-used software to capture, manage, and process within a tolerable elapsed time. This drives and motivates research and practices in *data science*—which aims to develop systematic or quantitative processes to analyze and mine big data—for continuous or iterative exploration, investigation, and understanding of past business performance so as to gain new insight and drive science or business planning. By applying *big data analytics and mining* (which incorporates various techniques from a broad range of fields such as cloud computing, data analytics, data mining, machine learning, mathematics, and statistics), data scientists can extract implicit, previously unknown, and potentially useful information from big data (e.g., big social network data).

Over the past few years, researchers have used a high-level programming model—called *MapReduce* [2]—to process high volumes of big data by using parallel and distributed computing on large clusters or grids of nodes (i.e., commodity machines) or clouds, which consist of a master node and multiple worker nodes. As implied by its name, MapReduce involves two key functions: (i) the map function and (ii) the reduce function. Specifically, the input data are read, divided into several partitions (sub-problems), and assigned to different processors. Each processor executes the *map function* on each partition (sub-problem). The map function takes a pair of $\langle key_1, value_1\rangle$ and returns a list of $\langle key_2, value_2\rangle$ pairs as an intermediate result, where (i) key_1 and key_2 are keys in the same or different domains and (ii) $value_1$ and $value_2$ are the corresponding values in some domains. Afterwards, these pairs are shuffled and sorted. Each processor then executes the *reduce function* on (i) a single key key_2 from this intermediate result $\langle key_2, \text{list of } value_2\rangle$ together with (ii) the list of all values that appear with this key in the intermediate result. The reduce function "reduces"—by combining, aggregating, summarizing, filtering, or transforming—the list of values associated with a given key key_2 (for all k keys) and returns a single (aggregated or summarized) value $value_3$, where (i) key_2 is a key in some domains and (ii) $value_2$ and $value_3$ are the corresponding values in some domains. An advantage of using the MapReduce model is that users only need to focus on (and specify) these "map" and "reduce" functions—without worrying about implementation details for (i) partitioning the input data, (ii) scheduling and executing the program across multiple machines, (iii) handling machine failures, or (iv) managing inter-machine communication. Examples of MapReduce applications include the construction of an inverted index as well as the word counting of a document for data processing [2].

3 Our Data Analytics Solution for Mining "Following" Patterns from Big Social Network Data

In this section, we present our new big data analytics and mining solution—called **BigFoP**—which mines **Big** social network data for interesting "**Fo**llowing" **p**atterns using the MapReduce model.

3.1 "Following" Relationships in Big Social Networks

In social networking sites like Twitter, social entities (users) are linked by *"following" relationships* such as A→B indicating that a user A (i.e., *follower*) follows another user B (i.e., *followee*). Then, given a social network in which each social entity is *following* some other social entities, such a social network can be represented as a graph $G = (V, E)$ where (i) V is a set of vertices (i.e., social entities) and (ii) E is a set of directional edges connecting some of these vertices (i.e., "following" relationships). See Example 1.

Example 1. For illustrative purpose, let us consider a small portion of a big social network as shown in Fig. 1. It can be represented by $G = (V, E)$, where (i) set V of vertices = {Abel, Biel, Carlos, Desi, Elba, Fabio} and (ii) set E of edges = {⟨Abel, B⟩, ⟨Abel, E⟩, ⟨Biel, A⟩, ⟨Biel, C⟩, ⟨Biel, E⟩, ⟨Carlos, A⟩, ⟨Carlos, E⟩, ⟨Desi, B⟩, ⟨Desi, C⟩, ⟨Desi, E⟩, ⟨Elba, A⟩, ⟨Elba, B⟩, ⟨Elba, C⟩, ⟨Elba, D⟩, ⟨Fabio, A⟩, ⟨Fabio, B⟩, ⟨Fabio, C⟩, ⟨Fabio, E⟩}. Here, to avoid the confusion between followers and followees, we represent followers by their names and followees by their initials in these ⟨follower, followee⟩-pairs in the set E of edges. □

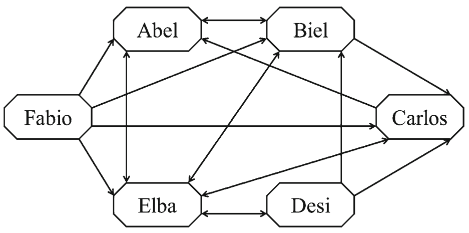


Fig. 1. A sample social network consisting of $|V| = 6$ users.

When compared with the mutual friendship relationships, the "following" relationships are different in that the latter are *directional.* For instance, a user B may be following another user C while C is not following B. As in Example 1, Biel is following Carlos, but Carlos is not following Biel. This property increases the complexity of the problem because of the following reasons. The group of users followed by B (e.g., Biel→{Abel, Carlos, Elba}) may not be same group of users as those who are following B (e.g., {Abel, Desi, Elba, Fabio}→Biel). Hence, we need to store directional edges (e.g., ⟨Abel, Biel⟩, ⟨Biel, Abel⟩) instead of undirectional edges (e.g., {Abel, Biel} indicating that Able and Biel are mutual friends). Given $|V|$ social entities, there are potentially $|V|(|V| - 1)$ directional edges for "following" relationships (cf. potentially $\frac{|V|(|V|-1)}{2}$ undirectional edges for mutual friendship relationships). Besides an increase in storage space, the computation time also increases because we need to check both directions to get relationships between pairs of users (e.g., cannot determine whether or not Carlos→Biel if we only know Biel→Carlos).

3.2 Discovery of "Following" Patterns

As the number of users in social networking sites (e.g., Twitter) is growing explosively nowadays, the number of "following" relationships between social network users is also growing. One of the important research problems with regard to this high volume of data is to discover interesting "following" patterns. A *"following" pattern* is a pattern representing the linkages when a significant number of users follow the same combination/group of users. For example, users who follow the twitter feed or tweets of NBA also follow the tweets of Adam Silver (current NBA commissioner). If there are large numbers of users who follow the tweets of both NBA and Adam Silver together, we can define this combination (NBA and Adam Silver) of followees as an interesting "following" pattern (i.e., a frequently followed group).

To discover interesting "following" patterns (i.e., collections of social network pages that are frequently followed by users), we propose a data science solution called **BigFoP** that mines **Big** social network data for interesting "**Fo**llowing" **p**atterns by using sets of map and reduce functions.

3.3 The First Set of Map-Reduce Functions in BigFoP

Abstractly, BigFoP first applies a map function to each edge as follows:

$$\begin{aligned}\text{map}_1 : &\langle \text{edge ID}, \text{“following” relationship captured by the edge}\rangle \\ &\mapsto \langle \text{ follower, individual followee}\rangle, \end{aligned} \tag{1}$$

in which the master node reads and divides big social network data in partitions. Specifically, the map_1 function can be specified as follows:

```
For each edge $e$=⟨follower, followee⟩ ∈ $E$ in social network $G$=$(V, E)$ do
    emit ⟨follower, followee, 1⟩.
```

This map function is applied to each edge e=⟨follower, followee⟩ $\in E$ in the social network represented by $G = (V, E)$, and results in a list of ⟨follower, followee, 1⟩ capturing all existing “following” relationships (between followers and followees) in the social network. See Example 2.

Example 2. After applying the map$_1$ function to the social network data in Example 1, our BigFoP returns a list containing ⟨Abel, B, 1⟩, ⟨Abel, E, 1⟩, ⟨Biel, A, 1⟩, ⟨Biel, C, 1⟩, ⟨Biel, E, 1⟩, ⟨Carlos, A, 1⟩, ⟨Carlos, E, 1⟩, ⟨Desi, B, 1⟩, ⟨Desi, C, 1⟩, ⟨Desi, E, 1⟩, ⟨Elba, A, 1⟩, ⟨Elba, B, 1⟩, ⟨Elba, C, 1⟩, ⟨Elba, D, 1⟩, ⟨Fabio, A, 1⟩, ⟨Fabio, B, 1⟩, ⟨Fabio, C, 1⟩, and ⟨Fabio, E, 1⟩. □

Afterwards, our big data analytics and mining solution BigFoP applies a reduce function to group and count the number of followers for each followee, as well as to list these followers for each followee. More specifically, ⟨follower, followee, 1⟩ pairs from the map$_1$ function are shuffled and sorted. Each processor then executes the reduce function on the shuffled and sorted pairs to count the number of followers and list them for each followee. To speed up this big social network data mining process, BigFoP also allows users to specify the *interestingness* of groups of social entities by a frequency threshold. Here, the users can indicate the minimum number of followers for a group of followees so that the group can be considered interesting. By incorporating this user preference, BigFoP returns (i) a list of followers only for those popular followees (i.e., followees who are frequently followed by at least the minimum number of followers) and (ii) the count for each followee. In other words, BigFoP applies the following reduce function:

$$\begin{aligned} \text{reduce}_1\!:&\langle \text{followee, list of followers}\rangle \\ &\mapsto \text{list of}\,\langle \textit{interesting}\ \text{followee, follower information}\rangle, \end{aligned} \tag{2}$$

with a detailed definition as follows:

```
For each followee ∈ ⟨_, followee, _⟩ emitted by map$_1$ do
    set counter[followee] = 0;
    set list[followee] = {};
    for each follower ∈ ⟨follower, followee, 1⟩ emitted by map$_1$ do
        counter[followee] = counter[followee] + 1;
        list[followee] = list[followee] ∪ {follower};
    if counter[followee] $\geq$ user-specified min frequency threshold
    then emit ⟨followee, counter[followee], list[followee]⟩.
```

This results in (i) a list of followers and (ii) its count for each *interesting/popular* followee. See Example 3.

Example 3. Continue with Example 2. Our BigFoP applies the reduce$_1$ function with user-specified minimum frequency threshold of 2 followers and returns ⟨A, 4, {Biel, Carlos, Elba, Fabio}⟩, ⟨B, 4, {Abel, Desi, Elba, Fabio}⟩, ⟨C, 4, {Biel, Desi, Elba, Fabio}⟩, and ⟨E, 5, {Abel, Biel, Carlos, Desi, Fabio}⟩. Note that our BigFoP does not return the lists for followees D or F because their corresponding counters were low (D and F were followed by only 1 and 0 followers, respectively).

To summarize, after applying the first set of map_1 and reduce_1 functions, our BigFoP has so far discovered four interesting "following" patterns—in the form of *individual* frequently followed social entities—namely, {A}, {B}, {C} and {E}, who are followed by 4, 4, 4 and 5 followers respectively. In other words, each of these four individual followees is followed by at least 2 followers (the user-specified minimum frequency threshold). □

3.4 The Second Set of Map-Reduce Functions in BigFoP

Thereafter, our BigFoP applies a next set of map and reduce functions to mine interesting "following" patterns in the form of *pairs* of frequently followed social entities based on the results from the first set of map_1 and reduce_1 functions. For instance, knowing that D and E are unpopular individual followees, it is guaranteed that any pairs containing followee D or E is also unpopular. By making use of this knowledge, the search space for mining interesting "following" patterns can then be pruned effectively. Specifically, the map_2 function, which returns $\langle$follower, $\{p\} \cup$ {followee}, 1$\rangle$ for every follower in the follower list of each popular/interesting individual followee p, can be specified as follows:

$$\begin{aligned}\text{map}_2 : &\langle \text{interesting followee } p, \text{its follower information}\rangle \\ &\mapsto \langle \text{follower, followee pair}\rangle, \end{aligned} \tag{3}$$

with a detailed definition as follows:

```
**For each** $p \in \langle p, \_, \text{list}[p]\rangle$ emitted by $\text{reduce}_1$ **do**
    **for each** follower $\in \text{list}[p]$ **do**
        **for each** $\langle$follower, followee$\rangle \in E$ of social network $G$=$(V, E)$ **do**
            **if** isRelevant(followee, $p$)
            **then emit** $\langle$follower, $\{p\} \cup$ {followee}, 1$\rangle$.
```

Here, isRelevant(followee, p) is a Boolean function checking the relevance (e.g., consistence to the mining order) of followee with respect to p. This results in lists of $\langle$follower, followee, 1$\rangle$, and a list for each popular individual followee p returned by the reduce_1 function. See Example 4.

Example 4. Continue with Example 3. Recall that the first set of map_1 and reduce_1 functions returns four popular followees A, B, C and E. So, for popular followee A (followed by four followers Biel, Carlos, Elba and Fabio), the map_2 function emits all *relevant* followees of these four followers: ⟨Biel, AC, 1⟩, ⟨Biel, AE, 1⟩, ⟨Carlos, AE, 1⟩, ⟨Elba, AB, 1⟩, ⟨Elba, AC, 1⟩, ⟨Fabio, AB, 1⟩, ⟨Fabio, AC, 1⟩, and ⟨Fabio, AE, 1⟩. Note that (i) followees of Abel are not emitted (because it is not meaningful for Abel to follow himself), (ii) followees of Desi are not emitted (because Desi does not follow A), (iii) four relationships in the form ⟨_, A, 1⟩ (e.g., ⟨Biel, A, 1⟩) are irrelevant with respect to p=A (because we already knew these four followers are following *single individual followee* A when we started this map_2 function and we aimed to find followers who follow *pairs of followees*), and (iv) ⟨Elba, AD, 1⟩ is also irrelevant (because followee D is unpopular).

Similarly, for popular followee B (followed by four followers Abel, Desi, Elba and Fabio), the map$_2$ function emits all *relevant* followee of these four followers: $\{\langle$Abel, BE, 1$\rangle$, $\langle$Desi, BC, 1$\rangle$, $\langle$Desi, BE, 1$\rangle$, $\langle$Elba, BC, 1$\rangle$, $\langle$Fabio, BC, 1$\rangle$, $\langle$Fabio, BE, 1$\rangle\}$. Note that (i) followees of Biel are not emitted (because it is not meaningful for Biel to follow himself), (ii) followees of Carlos are not emitted (because Carlos does not follow B), (iii) four relationships in the form $\langle$_, B, 1$\rangle$ (e.g., $\langle$Desi, B, 1$\rangle$) are irrelevant with respect to p=B (because we already knew these four followers are following *single individual followee* B when we started this map$_2$ function and we aimed to find followers who follow *pairs of followees*), and (iv) $\langle$Elba, BD, 1$\rangle$ is also irrelevant (because followee D is unpopular). More important to note is that (v) relationships in the form $\langle$_, AB, 1$\rangle$ (e.g., $\langle$Elba, AB, 1$\rangle$, $\langle$Fabio, AB, 1$\rangle$) are irrelevant with respect to p=B (because these relationships are already processed by the map$_2$ function).

Then, for popular followee C (followed by four followers Biel, Desi, Elba and Fabio), the map$_2$ function emits all *relevant* followee of these four followers: $\{\langle$Biel, CE, 1$\rangle$, $\langle$Desi, CE, 1$\rangle$, $\langle$Fabio, CE, 1$\rangle\}$.

Finally, for popular followee E (followed by five followers Abel, Biel, Carlos, Desi and Fabio), the map$_2$ function does not emit any followee because there is no *relevant* followee for these five followers. □

Similar to reduce$_1$, the reduce$_2$ function shuffles and sorts $\langle$follower, $\{p\} \cup$ {relevant followee}, 1$\rangle$ to find and count followers for each followee pair P = ($\{p\} \cup$ {relevant followee}) as follows:

$$\begin{aligned} &\text{reduce}_2 : \langle \text{followee pair, list of common followers} \rangle \\ &\qquad \mapsto \text{list of } \langle \textit{interesting} \text{ followee pair, follower information} \rangle, \end{aligned} \tag{4}$$

with a detailed definition as follows:

```
For each $P \in \langle$_, followee group $P$, _$\rangle$ emitted by map$_2$ do
    set counter[$P$] = 0;
    set list[$P$] = {};
    for each follower $\in \langle$follower, $P$, 1$\rangle$ emitted by map$_2$ do
        counter[$P$] = counter[$P$] + 1;
        list[$P$] = list[$P$] $\cup$ {follower};
    if counter[$P$] $\geq$ user-specified min frequency threshold
    then emit $\langle P$, counter[$P$], list[$P$]$\rangle$.
```

This results in (i) a list of followers and (ii) its count for each *interesting/popular* followee pair P.

Example 5. Continue with Example 4. Our BigFoP applies the reduce$_2$ function with user-specified minimum frequency threshold = 2 followers and returns $\langle$AB, 2, {Elba, Fabio}$\rangle$, $\langle$AC, 3, {Biel, Elba, Fabio}$\rangle$, $\langle$AE, 3, {Biel, Carlos, Fabio}$\rangle$, $\langle$BC, 3, {Desi, Elba, Fabio}$\rangle$, $\langle$BE, 3, {Abel, Desi, Fabio}$\rangle$, and $\langle$CE, 3, {Biel, Desi, Fabio}$\rangle$. In other words, after applying this second set of map$_2$ and reduce$_2$ functions, our BigFoP algorithm discovered six interesting "following" patterns—in the form of pairs of frequently followed social entities—namely, {A,B}, {A,C}, {A,E}, {B,C}, {B,E} and {C, E}, who are followed by 2, 3, 3, 3, 3 and 3 followers respectively. In other words, each of these six followee pairs is followed by at least 2 followers (the user-specified minimum frequency threshold). □

3.5 Subsequent Sets of Map-Reduce Functions in BigFoP

So far, our BigFoP has found interesting "following" patterns in the form of (i) *individual* frequently followed social entities as well as (ii) *pairs* of frequently followed social entities. BigFoP then applies *similar* sets of map and reduce functions to find triplets, quadruplets, quintuplets and higher (i.e., k-tuplets for $k \geq 3$) of frequently followed social entities:

$$\begin{aligned}\text{map}_{k\geq3} : &\langle\text{interesting followee}(k-1)\text{-tuplet } P\text{, its follower information}\rangle\\ &\mapsto \langle\text{follower, followee } k\text{-tuplet}\rangle, \end{aligned} \tag{5}$$

with a detailed definition as follows:

```
For each $P \in \langle P, \_, \text{list}[P]\rangle$ emitted by $\text{reduce}_{k-1}$ do
    for each follower $\in \text{list}[P]$ do
        for each $\langle$follower, followee$\rangle \in E$ of social network $G=(V,E)$ do
            if isRelevant(followee, $P$)
            then emit $\langle$follower, $P \cup \{$followee$\}$, 1$\rangle$.
```

Again, isRelevant(followee, P) is a Boolean function checking the relevance (e.g., consistence to the mining order) of followee with respect to P. Since reduce$_2$ can be considered as an instance of the reduce$_{k\geq2}$ function, the latter can be defined in a way very similar to that for reduce$_2$ as shown below:

$$\begin{aligned}\text{reduce}_{k\geq2} : &\langle\text{followee group, list of common followers}\rangle\\ &\mapsto \text{list of } \langle\textit{interesting}\text{ followee group, follower information}\rangle, \end{aligned} \tag{6}$$

with a detailed definition as follows:

```
For each $P \in \langle \_, \text{followee group } P, \_\rangle$ emitted by $\text{map}_{k-1}$ do
    set counter[$P$] = 0;
    set list[$P$] = {};
    for each follower $\in \langle$follower, $P$, 1$\rangle$ emitted by $\text{map}_{k-1}$ do
        counter[$P$] = counter[$P$] + 1;
        list[$P$] = list[$P$] $\cup$ {follower};
    if counter[$P$] $\geq$ user-specified min frequency threshold
    then emit $\langle P$, counter[$P$], list[$P$]$\rangle$.
```

This results in (i) a list of followers and (ii) its count for each *interesting/popular* followee group P. See Example 6.

Example 6. Continue with Example 5. For popular followee group AB (followed by two followers Elba and Fabio), the map$_3$ function emits three *relevant* followees: {⟨Elba, ABC, 1⟩, ⟨Fabio, ABC, 1⟩, ⟨Fabio, ABE, 1⟩}. Then, for popular followee group AC (followed by three followers Biel, Elba and Fabio), the map$_3$ function emits two *relevant* followees: $\{\langle$Biel, ACE, 1$\rangle$, $\langle$Fabio, ACE, 1$\rangle\}$. Similarly, for popular followee group BC (followed by three followers Desi, Elba and Fabio), the map$_3$ function emits two *relevant* followees: $\{\langle$Desi, BCE, 1$\rangle$, $\langle$Fabio, BCE, 1$\rangle\}$.

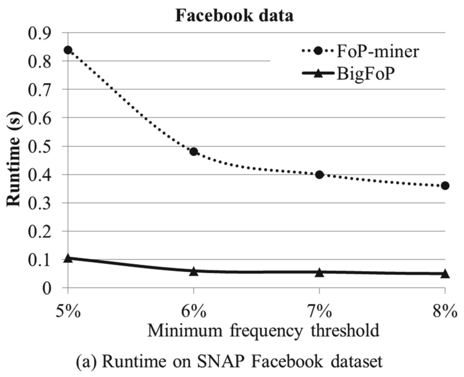
Afterwards, by applying the reduce$_3$ function, our BigFoP discovers the following three interesting "following" patterns {A, B, C}, {A, C, E} and {B, C, E} with their associated lists and number of followees as $\langle$ABC, 2, {Elba, Fabio}$\rangle$, $\langle$ACE, 2, {Biel, Fabio}$\rangle$, and $\langle$BCE, 2, {Desi, Fabio}$\rangle$.

Based on the results returned by the reduce$_3$ function, BigFoP applies map$_4$ but returns nothing because there is no relevant quadruplet of frequently followed social entities. This completes the mining process for interesting "following" patterns from our illustrative example social network. Note that key concepts and steps illustrated in this example are applicable to any big social network. □

4 Observations, Evaluation and Discussion

To discover "following" patterns, our BigFoP takes advantages of the MapReduce model. The input social data are divided into several partitions (subproblems) and assigned to different processors. Each processor executes the map$_k$ and reduce$_k$ functions (for $k \geq 1$). On the surface, one might worry that lots of communications or exchanges of information are required among processors. Fortunately, due to the divide-and-conquer nature of our big social network data analytics solution of discovering "following" patterns, once the original big social network is partitioned and assigned to each processor (e.g., one processor is assigned the followers of A, another is assigned the followers of B, a third one is assigned the followers of C), each processor handles the assigned data without any reliance on the results from other processors. As observed from the above examples, the processor assigned for the followers of a popular followee can apply the subsequent sets of map and reduce functions on data emitted by that processor. For example, a processor applies map$_1$ and reduce$_1$ to find popular followee A. That processor can then apply map$_2$ on the data emitted by reduce$_1$ from that processor to find popular followee group AB (i.e., group containing A). Similarly, the processor applies map$_3$ on the data emitted by reduce$_2$ from the same processor to find subsequent popular followee group ABC. Without the need of extra communications and exchanges of data among processors, our BigFoP discovers all interesting "following" patterns efficiently. Moreover, if a partition of the big social network is too big to be handled by a single processor, our BigFoP furthers sub-divide that partition so that the resulting sub-partitions can be handled by each of the multiple processors.

Furthermore, due to the divide-and-conquer nature of our big social network data analytics solution of discovering "following" patterns, the amount of data input for the map$_k$ and reduce$_k$ functions monotonically decreases as the size of the popular group of k followees increases. Our BigFoP discovers all interesting "following" patterns in a space effective manner.



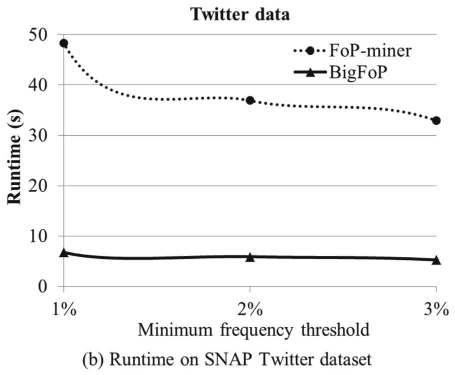


Fig. 2. Experimental results of BiigFoP on social network datasets.

As for runtime performance, we compared the performance of our BigFoP with related works (e.g., FoP-miner [4]). We used real-life social network datasets: The Stanford Network Analysis Project (SNAP) ego-Facebook dataset and ego-Twitter dataset (http://snap.stanford.edu/data/). The SNAP Facebook dataset contains 4,039 social entities and 88,234 connections ("following" relationships) between these social entities. The SNAP Twitter dataset contains 81,306 social entities and 1,768,149 connections between these social entities. All experiments were run using either (i) a single machine with an Intel Core i7 4-core processor (1.73 GHz) and 8 GB of main memory running a 64-bit Windows 7 operating system, or (ii) the Amazon Elastic Compute Cloud (EC2) cluster—specifically, 11 High-Memory Extra Large (m2.xlarge) computing nodes (http://aws.amazon.com/ec2/). We implemented both the existing FoP-miner algorithm and our proposed BigFoP in the Java programming language. The stock version of Apache Hadoop 0.20.0 was used. The results shown in Fig. 2, in which the x-axis shows the user-specified minimum frequency threshold (in percentage of the number of social entities) expressing the interestingness of the mined patterns, are based on the average of multiple runs. Runtime includes CPU and I/Os in the mining process of interesting "following" patterns. In particular, Fig. 2(a) shows that BigFoP provided a speedup of about 8 times when compared with FoP-miner when mining the SNAP Facebook dataset. Higher speedup is expected when using more processors. Figure 2(b) shows a similar result for the SNAP Twitter dataset. Moreover, our BigFoP is shown to be scalable with respect to the number of social entities in the big social network. As ongoing work, we are conducting more experiments, including an in-depth study on the quality of discovered "following" patterns.

5 Conclusions

In this paper, we proposed a big data analytics and mining algorithm—called *BigFoP*—for discovering interesting "following" patterns. BigFoP helps social network users to discover groups of frequently followed followees from big social networks by using the MapReduce model. By applying BigFoP, social network users (e.g., newcomers) could find popular groups of followees and follow them. Similarly, a business could find popular groups of followed products and services and incorporate customers' feedback on these products and services. Experimental results show the effectiveness of BigFoP in this big data analytics task of mining social networks for interesting "following" patterns.

Acknowledgement. This project is partially supported by NSERC (Canada) and University of Manitoba.

References

1. Cuzzocrea, A., Leung, C.K.-S., MacKinnon, R.K.: Mining constrained frequent itemsets from distributed uncertain data. Future Gener. Comput. Syst. **37**, 117–126 (2014)
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
3. Dhahri, N., Trabelsi, C., Ben Yahia, S.: RssE-Miner: a new approach for efficient events mining from social media RSS feeds. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 253–264. Springer, Heidelberg (2012)
4. Jiang, F., Leung, C.K.-S.: Mining interesting "Following" patterns from social networks. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 308–319. Springer, Heidelberg (2014)
5. Jiang, F., Leung, C.K.-S.: Stream mining of frequent patterns from delayed batches of uncertain data. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 209–221. Springer, Heidelberg (2013)
6. Jiang, F., Leung, C.K.-S., Liu, D., Peddle, A.M.: Discovery of really popular friends from social networks. In: IEEE BDCloud 2014, pp. 342–349. IEEE, Los Alamitos (2014)
7. Kang, Y., Yu, B., Wang, W., Meng, D.: Spectral Clustering for Large-Scale Social Networks via a Pre-Coarsening Sampling based Nyström Method. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015, Part II. LNCS (LNAI), vol. 9078, pp. 106–118. Springer, Heidelberg (2015)
8. Leung, C.K.-S., Cuzzocrea, A., Jiang, F.: Discovering frequent patterns from uncertain data streams with time-fading and landmark models. LNCS TLDKS **8**, 174–196 (2013)
9. Leung, C.K.-S., MacKinnon, R.K.: BLIMP: a compact tree structure for uncertain frequent pattern mining. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 115–123. Springer, Heidelberg (2014)
10. Leung, C.K.-S., MacKinnon, R.K., Tanbeer, S.K.: Fast algorithms for frequent itemset mining from uncertain data. In: Kumar, R., Toivonen, H., Pei, J., Huang, J.Z., Wu, X. (eds.) IEEE ICDM 2014, pp. 893–898. IEEE, Los Alamitos (2014)
11. Leung, C.K.-S., Tanbeer, S.K.: Mining popular patterns from transactional databases. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 291–302. Springer, Heidelberg (2012)
12. Leung, C.K.-S., Tanbeer, S.K., Cameron, J.J.: Interactive discovery of influential friends from social networks. Soc. Netw. Anal. Min. **4**(1), Article 154 (2014)

13. Ma, L., Huang, H., He, Q., Chiew, K., Wu, J., Che, Y.: GMAC: a seed-insensitive approach to local community detection. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 297–308. Springer, Heidelberg (2013)
14. Madden, S.: From databases to big data. IEEE Internet Comput. **16**(3), 4–6 (2012)
15. Mumu, T.S., Ezeife, C.I.: Discovering community preference influence network by social network opinion posts Mining. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 136–145. Springer, Heidelberg (2014)
16. Rader, E., Gray, R.: Understanding user beliefs about algorithmic curation in the facebook news feed. In: Begole, B., Kim, J., Inkpen, K., Woo, W. (eds.) ACM CHI 2015, pp. 173–182. ACM, New York (2015)
17. Rajadesingan, A., Zafarani, R., Liu, H.: Sarcasm detection on Twitter: a behavioral modeling approach. In: Cheng, X., Li, H., Gabrilovich, E., Tang, J. (eds.) ACM WSDM 2015, pp. 97–106. ACM, New York (2015)
18. Tanbeer, S.K., Leung, C.K.-S., Cameron, J.J.: Interactive mining of strong friends from social networks and its applications in e-commerce. J. Organ. Comput. Electron. Commer. **24**(2–3), 157–173 (2014)
19. Wei, E.H.-C., Koh, Y.S., Dobbie, G.: Finding maximal overlapping communities. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 309–316. Springer, Heidelberg (2013)
20. Yu, W., Coenen, F., Zito, M., El Salhi, S.: Minimal vertex unique labelled subgraph mining. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 317–326. Springer, Heidelberg (2013)