Mining Frequent Patterns from Human Interactions in Meetings Using Directed Acyclic Graphs

Anna Fariha[1], Chowdhury Farhan Ahmed[1], Carson Kai-Sang Leung[2], S M Abdullah[3], and Longbing Cao[4]

[1]Department of Computer Science and Engineering, University of Dhaka, Bangladesh
purpleblueanna@gmail.com, farhan@khu.ac.kr
[2] University of Manitoba, Canada
kleung@cs.umanitoba.ca
[3] United International University, Bangladesh
smab@cse.uiu.ac.bd
[4] University of Technology - Sydney, Australia
longbing.cao@uts.edu.au

Abstract. In modern life, interactions between human beings frequently occur in meetings, where topics are discussed. Semantic knowledge of meetings can be revealed by discovering interaction patterns from these meetings. An existing method mines interaction patterns from meetings using tree structures. However, such a tree-based method may not capture all kinds of triggering relations between interactions, and it may not distinguish a participant of a certain rank from another participant of a different rank in a meeting. Hence, the tree-based method may not be able to find all interaction patterns such as those about correlated interaction. In this paper, we propose to mine interaction patterns from meetings using an alternative data structure—namely, a *directed acyclic graph* (*DAG*). Specifically, a DAG captures both temporal and triggering relations between interactions in meetings. Moreover, to distinguish one participant of a certain rank from another, we assign weights to nodes in the DAG. As such, a meeting can be modeled as a weighted DAG, from which weighted frequent interaction patterns can be discovered. Experimental results showed the effectiveness of our proposed DAG-based method for mining interaction patterns from meetings.

Key words: Data mining, Frequent patterns, Human interaction, Modeling meetings, Directed Acyclic Graphs.

1 Introduction and Related Works

Meetings and human interactions are integral parts of workplace dynamics for communicating between members participating in a meeting. During a meeting, several kinds of human interactions may occur. Examples include (i) proposing an idea, (ii) positively or negatively reacting to a proposal, (iii) acceptance of

a proposal. To gather significant information regarding the success rate of the decision made in a meeting, one can mine patterns from human interactions occurred in the meeting.

Data mining is useful in discovering implicit, previously unknown, and potentially valuable information or knowledge from large datasets. For instance, frequent pattern mining [1, 4, 7] is helpful in finding frequently occurring patterns, such as *interaction patterns* from meetings. The discovered interaction patterns help to (i) estimate the effectiveness of decisions made in meetings, (ii) designate whether a meeting discussion is fruitful, (iii) compare two meeting discussions using interaction flow as a key feature [12] and (iv) index meetings for further ease of access in database.

To acquire the semantic information from a meeting, researchers extracted the meeting contents and represented them in a machine readable format. For instance, Waibel et al. [9] presented a meeting browser that describes the dynamics of human interactions. McCowan et al. [5] recognized group actions in meetings by modeling the joint behavior of participants and expressed group actions as a two-layer process by a hidden Markov model framework. Otsuka et al. [6] used gaze, head gestures, and utterances to determine who responds to whom in multiparty face-to-face conversations. Yu et al. [11] proposed a multimodal approach for interaction recognition; they [12] also used a tree-based mining method to discover frequent patterns from human interactions occurred in meetings. Such a method focuses mostly on capturing direct parent-child relations. However, there are other triggering relations in meetings as illustrated in Example 1.

Example 1. Let us consider a scenario about a meeting of four persons (e.g., professor A, assistant professor B, and two lecturers C & D) with different weights/ranks. At the beginning of the meeting, B proposes an idea which triggers three interactions: (i) C expresses his negative opinion towards the proposed idea, (ii) C asks D for opinion on the idea, and (iii) A expresses some positive opinion towards the idea. Now, the interaction of C's request for D's opinion triggers a single interaction performed by D. Although the response of D is triggered by C's request of opinion, such a response is generally influenced by A's positive opinion. To elaborate, D may initially feel negatively regarding B's proposed idea. But, after listening to A's positive comments, B may change his mind and lean towards a neutral or even positive opinion. □

Example 1 reveals that (i) an interaction can be triggered or influenced by multiple interactions and (ii) the extent of influence can be significantly dependent on the weight/rank of the person triggering that interaction. However, the aforementioned tree-based method [12] does not capture these triggering relations. Moreover, as this method does not associate the actions with the rank of the person causing the actions, it does not distinguish the same kinds of actions performed by two persons having different weights/ranks.

Observing that (i) directed acyclic graphs (DAGs) and trees are both specializations of graphs and (ii) trees may not capture all triggering relations or

person's weight/rank, we explore the use of DAGs (as alternatives to trees) for modeling meetings. As interactions occur in meetings flow in only one direction with respect to time (i.e., no cycle), DAGs would be a logical choice for modeling meetings. A *key contribution* of this paper is our DAG-based modeling of interactions occurred in meetings. In particular, *DAGs* capture two kinds of relations: (i) temporal relations and (ii) triggering relations (cf. *trees* capture temporal relations but not all triggering relations). By doing so, each interaction is represented by a node in a DAG, and the label of the node indicates the class of interaction. Moreover, every node is associated with a *weight*, indicating the rank of the person who initiates the interaction.

Another *key contribution* of this paper is our DAG-based mining of weighted frequent interaction patterns from meetings. Note that, Chen et al. [3] mined DAG patterns from DAG databases. Termier et al. [8] presented DigDag as the first algorithm to mine closed frequent embedded sub-DAGs. Werth et al. [10] designed and implemented a DAG miner for mining DAGs from DAG databases. However, these related works do not consider weights of nodes in DAGs, let alone mining *weighted* frequent interaction patterns. Furthermore, there exist related works [2] that mine weighted frequent patterns from *transactional databases* (cf. *DAGs*). Inspired by these works, we integrate DAG-based mining and interaction pattern mining [10] with weighted frequent pattern mining [2] to form our **WDAGmeet** algorithm for **W**eighted **DAG**-based **meet**ing mining.

The rest of this paper is organized as follows. Section 2 introduces our DAG-based representation of interaction flow in meetings. Section 3 presents our weighted DAG-based frequent interaction pattern mining. Evaluation results are shown in Section 4, and conclusions are given in Section 5.

2 DAG-Based Representation of Interaction Flow

In this section, we introduce a DAG-based representation of interaction flow in decision-making meetings. Human interactions occurred in these meetings can be mainly categorized into the following nine classes:

1. **PRO:** A participant proposes an idea.
2. **ASK:** A participant asks for opinion regarding a proposal.
3. **POS:** A participant expresses positive attitude towards a proposal.
4. **NEG:** A participant expresses negative attitude towards a proposal.
5. **ACK:** A participant agrees on some other's comment, decision, or attitude.
6. **COM:** A participant comments on another action (PRO, ACK, POS, etc.).
7. **REQ:** A participant requests information regarding an issue.
8. **ACC:** A participant accepts the proposed idea.
9. **REJ:** A participant rejects the proposed idea.

When building a DAG to model the interaction flow occurred in meetings, we label each node in the DAG with one of the above nine classes of human interactions.

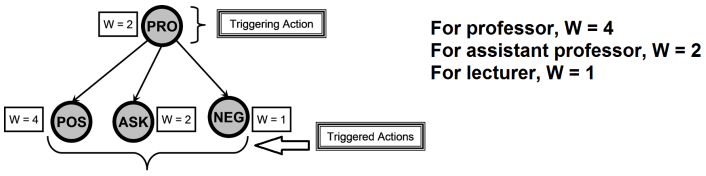

Fig. 1. A DAG-based representation of interaction flow showing triggering relations.

To further specify the rank of the person who initiated the interaction, we assign a weight with value ranging from 1 to n inclusive (e.g., n=3, 4 or 5). Although the same response can be made from different persons of different ranks, a response from a person having a heavier weight usually strongly influences the decision making process than that from a person of a lighter weight. Each node in the weighted DAG in Fig. 1 denotes an instance of human interaction occurred in a meeting. The label and weight of the nodes indicate the class of the interaction and the corresponding impact factor, respectively.

So far, we have categorized interactions into nine classes (e.g., PRO, ASK) based on the activities in a meeting. From the perspective of spontaneity, these interactions can also be categorized into two types of interactions: (i) *triggering interaction* and (ii) *triggered interaction*. For example, the PRO node in Fig. 1 reflects a triggering interaction, which represents an assistant professor proposes an idea spontaneously. The remaining nodes (POS, ASK & NEG) reflect three triggered interactions, which occur in response to the triggering interaction. Directed edges between nodes indicate **triggering relations** between the nodes, and the arrows point from the triggering interaction to the triggered one. Consequently, we generate a DAG-based interaction flow diagram for modeling meetings.

Besides those triggering relations, our proposed DAG-based representation of interaction flow also captures **temporal relations**. To elaborate, a DAG represents the temporal relations by topological level. Nodes of a certain topological level appear temporally before nodes of the next/lower level. Within the same topological level, the node on the left appears temporally before the node on the right. See Example 2.

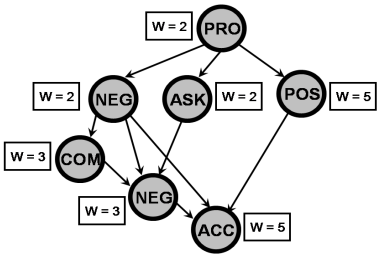

Fig. 2. A DAG-based representation of interaction flow showing both triggering relations and temporal relations in a meeting.

Example 2. Fig. 2 shows a sample session of a meeting, in which an assistant professor A proposes an idea. Triggered by A's proposed idea, one of his colleagues B first expresses her negative opinion and then asks others' opinions. On the other hand, a professor C (with heavier weight) expresses his positive opinion on A's idea. An associate professor D first comments on B's negative opinions. Based on both his comments and B's negative opinion, D then expresses his negative opinion in response to B's asking of opinion. Finally, based on two negative opinion from assistant professor B and associate professor D as well as the positive opinion from professor C, professor E accepts A's idea, biased to the interaction performed by person of higher rank. Note that, Fig. 2 captures not only single triggering relations but also interactions triggered by multiple triggering interactions. □

3 DAG-Based Frequent Pattern Mining from Interaction Flow DAGs

Once the DAG-based interaction flow diagram is generated, we can mine frequent interaction patterns (sub-DAG patterns) from the diagram. Before describing the key steps in this interaction pattern mining process, let us consider the following definitions.

Definition 1 (DAG-based Interaction Flow). One single meeting may consist of several sessions. Interaction flow within each session can be represented by a DAG $D = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of n vertices and $E = \{e_1, e_2, ..., e_m\}$ is a set of m directed edges. All DAGs are connected acyclic graphs and no two DAGs, representing sessions of the same meeting, are connected to each other. Each node v_i is assigned a class label $L(v_i)$, where $L(v_i) \in$ {PRO, ASK, POS, NEG, ACK, COM, REQ, ACC, REJ}. Each node is associated with a weight $W(v_i)$ that carries information regarding the (absolute or relative) rank of a participant who initiates an interaction in a meeting. Each edge is a directed connection between two vertices, i.e., $E = \{(v_i, v_j) | 1 \leq i, j \leq n; v_i, v_j \in V\}$. Here, v_i denotes the source/origin of the directed edge and v_j denotes the destination of that edge. An edge from v_i to v_j implies that v_i (completely or partially) triggers v_j. The levels of interactions can be determined according to their topological orders. Interactions of higher levels occur earlier than those of lower levels. Within the same level, interactions on the left occur earlier than those on the right. □

Definition 2 (Sub-DAG and Super-DAG). Consider two DAGs $D = (V, E)$ and $D' = (V', E')$ such that (i) D' is a connected DAG; (ii) $V' \subseteq V$; (iii) $E' \subseteq E$; (iv) for each $v'_i \in V', L(v'_i) = L(v_i)$ and $W(v'_i) = W(v_i)$ for a $v_i \in V$; (v) for each $e = (v'_i, v'_j) \in E'$, these v'_i & $v'_j \in V'$ are mapped to the corresponding v_i & $v_j \in V$. Then, D' is a **sub-DAG** of D. Equivalently, D is a **super-DAG** of D'. □

Definition 3 (Support). Given (i) a sub-DAG D' and (ii) a database DB, the support of D' is defined by the following equation:

$$sup(D') = \frac{\#\text{superDAGs of } D' \times \text{ avg weight of nodes in } D'}{\#\text{DAGs in } DB \times \text{ max weight of nodes in } DB}. \quad (1)$$

This definition of support allows us to discover sub-DAGs containing nodes that are not too frequent but are associated with heavy weights. See Example 3.

Example 3. Consider a sample DAG D, consisting of 10 directed edges (i.e., 10 triggering relations) on 20 nodes (i.e., 20 interactions): Professor A proposes three ideas (PRO), and each of them are rejected (REJ) by Professor B. Lecturer C makes 7 comments (COM), and each of them triggers Lecturer F's comments (COM). In other words, $D = (V, E)$, where (i) $V = \{v_1, ..., v_{20}\}$, (ii) $L(v_1) = L(v_3) = L(v_5) =$ PRO, (iii) $L(v_2) = L(v_4) = L(v_6) =$ REJ, (iv) $L(v_7) = ... = L(v_{20}) =$ COM, (v) $E = \{(v_1, v_2), (v_3, v_4), ..., (v_{19}, v_{20})\}$, (vi) $W(v_1) = ... = W(v_6) = 5$, and (vii) $W(v_7) = ... = W(v_{20}) = 1$. Here, the frequency of the pattern "A proposes an idea, which is rejected by B" is 3; the frequency of another pattern "C makes a comment, which is commented by F" is 7. Between them, the first pattern is more interesting than the second one because interactions between persons of higher rank (i.e., heavier weights) are usually more important and useful in analyzing decision-making meetings even when the frequency of these interactions is not too high. □

Definition 4 (Frequent Pattern or Fragment). Sub-DAGs, having support greater than the user-specific minimum support threshold *minsup* are considered **frequent sub-DAG pattern (or fragment)**. □

Definition 5 (Mining Frequent Interaction Patterns from Meeting DB). Given (i) a meeting database DB capturing human interactions in meetings, (ii) a user-specific minimum support threshold *minsup*, the problem of mining frequent interaction patterns is to discover from DB every frequent interaction pattern, i.e., every sub-DAG D' having $sup(D') \geq minsup$. □

Our proposed weighted DAG-based meeting mining algorithm (**WDAG-meet**) discovers frequent interaction patterns in the form of frequent sub-DAGs from weighted DAG database DB as follows. The algorithm first generates a set of all frequent nodes in DB. It then expands these nodes (i.e., singleton sub-DAGs) using the following four expansion rules:

1. **New Root:** A new root (with no incoming edge) is inserted.
2. **New Level:** A new topological level is introduced with the insertion of a new node into that level and the insertion of an edge from a node in the previous topological level.
3. **New Node:** A new node (with a label lexicographically greater than the last node in current topological level) is inserted into the current topological level.

Algorithm 1: WDAGmeet

Input : (1) Meeting DB of DAG, (2) $minsup$ threshold
Output: A set F of frequent DAG interaction patterns

begin
  $F \leftarrow$ all the frequent nodes in all the DAGs in DB
  $tmpF \leftarrow F$
  while $tmpF \neq \emptyset$ **do**
    $tmp \leftarrow \emptyset$
    for $f \in tmpF$ **do**
      $tmp \leftarrow tmp \cup expandWithRoots(DB, f, minsup)$
      $tmp \leftarrow tmp \cup expandWithNewLevel(DB, f, minsup)$
      $tmp \leftarrow tmp \cup expandWithNewNode(DB, f, minsup)$
      $tmp \leftarrow tmp \cup expandWithNewEdge(DB, f, minsup)$
    end for
    $tmp \leftarrow pruneNonCannonical(tmp, minsup)$
    $tmp \leftarrow filterNotWantedFragments(tmp, minsup)$
    $tmp \leftarrow filterNotFrequentFragments(tmp, minsup)$
    $F \leftarrow F \cup findConnectedDAGs(tmp)$
    $tmpF \leftarrow tmp$
  end while
end

4. **New Edge:** A new edge from a previously inserted node to the most recently inserted node is added.

See Example 4 for illustration of these rules. Note that, these rules are designed in such a way that no duplicate DAG is generated. In the process of expansion of already found frequent sub-DAGs, duplicate sub-DAGs may be generated. Although these duplicates do not affect the mining result, they certainly increase the runtime of the algorithm. To avoid generating duplicates, all newly expanded sub-DAGs are checked for duplicate canonical form [10] because the canonical form is unique for all duplicate isomorphic DAGs.

Example 4. Consider Fig. 3, which shows some examples of applications of the four expansion rules: WDAGmeet algorithm (a) inserts a *new root* PRO, (b) inserts another new root REQ, and (c) introduces a *new level* with the insertion of POS and of a triggering relation from PRO to POS. Afterwards, WDAGmeet (d) inserts into the current level a *new node* NEG triggered by REQ, (e) adds a *new edge* from PRO to the most recently inserted NEG. Similarly, WDAGmeet (f) introduces another new level (with the insertion of NEG, to which a triggering relation from another NEG is inserted) and (g) adds a new edge from PRO. □

Note that, some of the expanded DAGs are connected, but some are not. WDAGmeet algorithm only inserts frequent *connected* DAGs to the mining result. The algorithm repeats the above expansion process until no new expansion

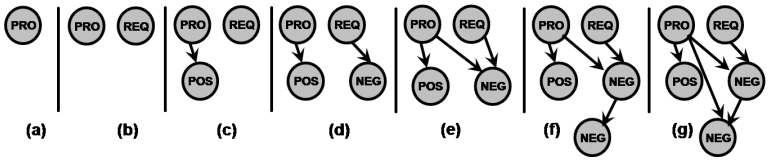

Fig. 3. Applications of expansion rules.

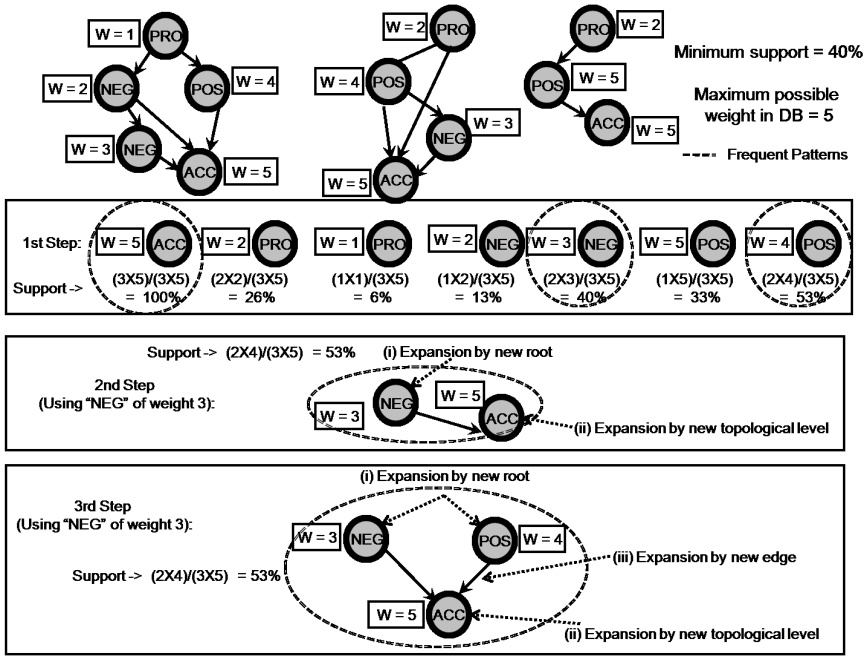

Fig. 4. First few steps of WDAGmeet algorithm.

is impossible. The pseudocode is given in Algorithm 1, and Fig. 4 illustrates the first few steps of the algorithm.

One important observation on the WDAGmeet algorithm is that, when the patterns are expanded, it adds not only frequent nodes but all possible nodes. The reason is that, the expansion rules do not satisfy the anti-monotone property: A pattern f may not be frequent because of low average-weight of the nodes contained in it, but connecting some nodes (of heavier weight or high support) can make f frequent.

4 Evaluation Results

First, we evaluated the functionality of our proposed WDAGmeet algorithm by comparing it with the existing tree-based mining method [12]. The tree-based

method misses some important frequent patterns because it does not capture all triggering relations. As illustrated in Fig. 5, only one triggering relation is captured in the tree database for each triggered interaction. For instance, the tree captures the interaction ASK triggered by PRO but misses the one triggered by POS. Similarly, the tree captures the interaction NEG triggered by PRO but misses the one triggered by ASK. As such, the tree-based method does not generate the pattern POS-ASK-NEG as these three nodes are *not directly* connected in the tree. In fact, fragments containing siblings or ancestor's siblings of a node might not be connected without the absence of their common ancestor in a tree. Hence, if the common ancestor is not frequent, the tree mining method fails to mine such fragments as a frequent pattern. In contrast, being internally connected with partial triggering relations, WDAGmeet discovers this kind of frequent interaction patterns, such as POS-ASK-NEG in the above example. This kind of frequent patterns reveals highly correlated interactions.

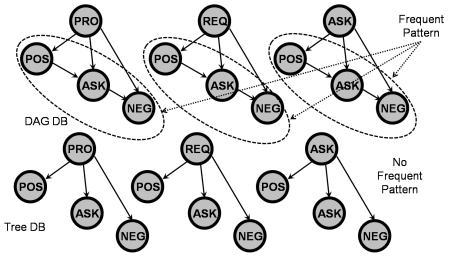

Fig. 5. DAG-based vs. tree-based representations of meetings.

Next, we evaluated empirically the performance and effectiveness of WDAGmeet algorithm, which was implemented in C++. In our experiments, we used datasets based on sample meetings. We generated 10 synthetic datasets to simulate real meeting scenarios. Each dataset contains a description of (i) the meeting captured in a DAG, (ii) labeled interactions with their corresponding weights, and (iii) triggering relations (i.e. directed edges of the DAG). We used five distinct weights for ranking each nine interaction with one of the nine class labels. Experiments were run using an Intel Core i5 2.50 GHz machine with 2.94 GB of RAM and 32 bit OS (Windows 7).

Table 1 shows the number of discovered frequent patterns and the elapsed time to discover these frequent patterns when using different minimum support *minsup* threshold values performed on 10 different datasets having different sizes. Table 1 shows that, as the number of frequent patterns increased, the required time to discover these frequent patterns also increased.

Fig. 6 plots the number of frequent patterns and elapsed time vs. DB size in the right and left, respectively. One can observe that, on average, the number of discovered frequent patterns (or fragments) was loosely related to the size of DAGs. Dense DAGs usually generated more frequent patterns than sparse ones.

Table 1. #frequent patterns & elapsed time with various *minsup* for different DB.

Size of DB (bytes)	*minsup* (%)	#frequent patterns	Elapsed time (seconds)
6917	40	415	440.589
	50	139	92.475
	60	26	17.672
	70	16	9.385
5804	30	481	401.539
	40	184	119.821
	50	25	8.917
	60	3	0.856
5465	50	135	86.668
	55	87	61.318
	60	73	53.787
	65	59	47.327

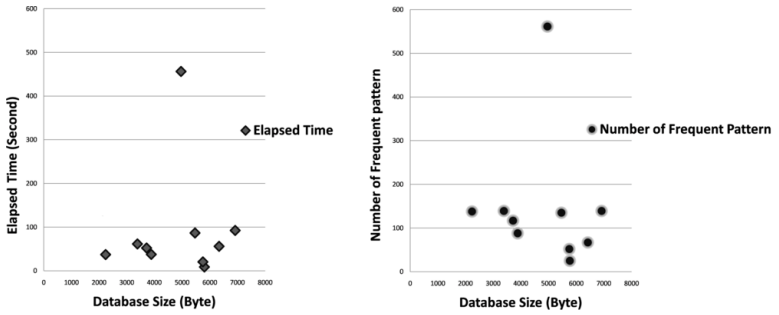

Fig. 6. Elapsed time and number of frequent patterns vs. DB size.

When patterns were generated, it was more likely to locate those patterns in a dense DAG than a sparse one because the dense DAG contains most of the probable edges. In contrast, the probability of finding a frequent pattern was low in a sparse DAG. The increment of DB size can partially represent the sparseness of the DAG capturing interactions in meetings.

Then, we compared the performance of WDAGmeet algorithm with that of the existing tree-based mining method [12] empirically. Table 2 shows our experimental results, which can be explained as follows. During the mining process, any frequent pattern must be connected because neither WDAGmeet nor the tree-based method can search DB for a pattern or fragment that is not connected. As discussed earlier, the tree-based mining method missed some frequent patterns. In contrast, WDAGmeet algorithm did not miss these patterns. Moreover, WDAGmeet algorithm used weighted nodes for representing the importance/rank of persons triggering each interaction. This criterion decreased the number of frequent patterns discovered by WDAGmeet. Moreover, WDAGmeet distinguished multiple interactions initiated by different persons having different weights. In contrast, the tree-based method did not distinguish mul-

tiple interactions. Hence, as WDAGmeet captures all triggering and temporal relations, it generated fewer frequent patterns and did not miss any frequent patterns. In contrast, the tree-based method generated more frequent patterns but also missed some frequent patterns.

Table 2. #frequent patterns & elapsed time for tree-based vs. our DAG-based mining.

Size of DB (bytes)	*minsup* (%)	#frequent patterns in tree-based method [12]	#frequent patterns in WDAGmeet
6917	40	635	415
	45	298	262
	50	153	139
5804	30	509	481
	35	482	442
	40	201	184
5465	50	146	135
	55	95	87
	60	79	73

To summarize, WDAGmeet algorithm discovered frequent interaction patterns from weighted DAGs capturing human interactions in meetings in reasonable amounts of time. When *minsup* increased, the number of discovered frequent patterns decreased and the elapsed time also decreased. When compared with the existing tree-based method (which captures few triggering and all temporal relations), WDAGmeet algorithm captures all triggering relations as well as all temporal relations. Moreover, WDAGmeet algorithm does not miss any frequent interaction patterns. As an ongoing work, we plan to conduct more extensive experiments and compare the precision, recall and F-measure of our proposed WDAGmeet algorithm with those of the existing tree-based method.

5 Conclusions

In this paper, we modeled human interactions in meetings using a weighted directed acyclic graph (DAG). The weight indicates the rank or importance of the person who initiates one of the nine classes of interactions. Such a DAG-based representation of interaction flow captures both (i) temporal relations and (ii) triggering relations (which connect the triggering interaction to the triggered interaction) in meetings. Moreover, we also proposed DAG-based frequent pattern mining from interaction flow DAGs. Specifically, our proposed WDAGmeet algorithm mines weighted DAG-based meeting for frequent interaction patterns. The key idea is to model each session (especially decision-making sessions) of a meeting using DAGs. Moreover, DAGs also include patterns or fragments that are connected without any common ancestor, previously missed by the existing tree-based method. The integration of weight assignment to each interaction makes the meeting mining process more robust and worthwhile.

Evaluation results show that WDAGmeet algorithm was more effective in discovering frequent interaction patterns from weighted DAGs than the existing tree-based method. The mined frequent sub-DAGs can be served as foundations to further association rule mining. As ongoing work, we plan to integrate other types of human interactions. Moreover, the resulting mining algorithm can be customized to handle other classes of meetings such as medical interviews and business discussions. The property of assigning weights to the interactions and preserving all kinds of partially triggering relations add functionality of WDAGmeet in mining patterns from human interactions in meetings.

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

References

1. Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.-K.: An efficient candidate pruning technique for high utility pattern mining. In: PAKDD 2009. LNAI 5476, pp. 749–756. Springer (2009)
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.-K., Choi, H.-J.: Single-pass incremental and interactive mining for weighted frequent patterns. Expert Systems with Applications 39(9), 7976-7994. Elsevier (2012)
3. Chen, Y.-L., Kao, H.-P., Ko, M.-T.: Mining DAG patterns from DAG databases. In: WAIM 2004. LNCS 3129, pp. 579–588 (2004)
4. Leung, C.K.-S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In: PAKDD 2008. LNAI 5012, pp. 653–661. Springer (2008)
5. McCowan, L., Gatica-Perez, D., Bengio, S., Lathoud, G., Barnard, M., Zhang, D.: Automatic analysis of multimodal group actions in meetings. IEEE TPAMI 27(3), 305–317 (2005)
6. Otsuka, K., Sawada, H., Yamato, J.: Automatic inference of cross-modal nonverbal interactions in multiparty conversations: "who responds to whom, when, and how?" from gaze, head gestures, and utterances. In: ICMI 2007, pp. 255–262. ACM (2007)
7. Tanbeer, S.K., Ahmed, C.F., Jeong, B.-S., Lee, Y.-K.: Discovering periodic-frequent patterns in transactional databases. In: PAKDD 2009. LNAI 5476, pp. 242–253. Springer (2009)
8. Termier, A., Tamada, Y., Numata, K., Imoto, S., Washio, T., Higuchi, T.: DIGDAG, a first algorithm to mine closed frequent embedded sub-DAGs. In: MLG (2007)
9. Waibel, A., Bett, M., Finke, M. Stiefelhagen, R: Meeting browser: tracking and summarizing meetings. In: DARPA Broadcast News Transcription and Understanding Workshop (1998)
10. Werth, T. Dreweke, A., Wörlein, M., Fischer, I., Philippsen, M.: DAGMA: mining directed acyclic graphs. In: IADIS ECDM 2008, pp. 11–18 (2008)
11. Yu, Z., Yu, Z., Ko, Y., Zhou, X., Nakamura, Y.: Inferring human interactions in meetings: a multimodal approach. In: UIC 2009. LNCS 5582, pp. 14–24 (2009)
12. Yu, Z., Yu, Z., Zhou, X., Becker, C., Nakamura, Y.: Tree-based mining for discovering patterns of human interaction in meetings. IEEE TKDE 24(4), 759–768 (2012)