CHAPTER 6

REFERRAL SYSTEMS

6.1 OVERVIEW

To solve some difficult problems that require procedural knowledge, users often seek the advice of experts, having competence in that problem domain. For example queries like “Which is the most suitable server available in the market to run high end multimedia applications?” can better be answered by a domain expert, having up-to-date knowledge of the server systems commercially available. Thus, the challenge of finding relevant information reduces to finding the right individual to answer the user queries. Social networks of the user are widely explored to locate such domain experts. Clearly, building and maintaining a central repository of social relationships is not feasible, because of considerations such as privacy. For this reason, distributed search through referrals is more promising by serving two key functions. First, it provides a reason for the expert to respond to the requester by explicating their relationship (friendship, kinship, collaborators etc.). Secondly, as the identities of individuals responding to the user queries are known to the user, it provides criterion to evaluate the trustworthiness of the expert.

This chapter focuses on the problem of finding domain experts through referrals in a time evolving co-author social network. Authors and co-authors of research publications for instance are domain experts. In a research domain, it is natural that experts may migrate from one knowledge
domain to another or young researchers may enter the domain and eventually become experts. However, static social networks widely studied in the literature may not effectively capture such scenario and also it would lead to periodic reconstruction of the entire network to include additions and deletions. The work in this chapter proposes a solution to capture the changing knowledge of researchers in a time evolving social network where the network is expanded incrementally, which can avoid periodic global expertise re-computation.

6.2 BASIC CONCEPTS AND DEFINITIONS

This study follows the conventional approach to represent the co-author social network as a graph. Formal definition of static and time evolving network is given below.

6.2.1 Static Co-Author Network

Definition 6.1 formally defines the static co-author network represented as an undirected graph $G = (V, E)$.

**Definition 6.1**

Let $V = \{v_i \mid i = 1 \text{ to } N\}$ be the set of authors, $P = \{p_i \mid i = 1 \text{ to } M\}$ be the set of papers, $a(p) = \{v_i \mid v_i \text{ is the author of } p \in P\}$ be the author set of a paper $p$ and $E = \{(v_i, v_j) \mid \text{if } v_i, v_j \in V \text{ and } v_i, v_j \in a(p) \text{ for some } p\}$.

The static co-author network thus defined is more appropriate to represent the linkages among the authors in a given time period and may not effectively study the evolving nature of this network.
6.2.2 Time Evolving Co-Author Network

The time evolving co-author social network, primarily studies two aspects of the data that may change over time. First, there may be relationships that vary over time. This results in a graph \( G_t = (V, E_t) \) at a given time \( t \), where the nodes remain constant but the edge set may vary, that is \( E_{t_i} \neq E_{t_j} \) for some time instance \( t_i, t_j \) as shown in Figure 6.1.

![Figure 6.1 Example of Co-Author Social Network where Edges Change over Time](image1)

![Figure 6.2 Example of Co-author Social Network where Nodes and Edges Change over Time](image2)
Secondly, there may be nodes and links that vary over time as shown in Figure 6.2. This study considers the scenario where the nodes and links changes over time. The time evolving co-author network shown in Figure 6.2 can be naturally defined as a sequence of graphs \( G_1, G_2, \ldots, G_t \). However, the information in a single time slice may not provide the complete publications and co-author list of any author. A simple and efficient way to model temporally-varying relational network like co-author network is to aggregate this sequence of co-author graphs. (Hill et al 2006).

### 6.3 TIME EVOLVING CO-AUTHOR NETWORK MODEL

As there are no real authors in the system, the referral system is built as a multi agent system where each author is associated with an agent that maintains their expertise and neighbor set model. In the subsequent discussion the terms author, user, expert and agent are used interchangeably. A query from the user is seen by the agent that suggests relevant people to whom the query can be sent. The relevant people are selected from the neighbor set model that initially includes only the immediate co-authors of an author and later can be expanded to include referrals. The neighboring agent that receives the query checks the model of its user, to decide if the query suits its user. If so, the query will be sent to the user, otherwise the agent may respond with referral or may decide to discard the query without responding to it, in consultation with its user.

Each agent is also associated with an expert cache, to include the experts found by it. Initially, the agents search the expert cache to locate experts relevant to the query. If the search is found to be successful, it sends the query directly to this expert instead of sending the query to its neighbors that can reduce the query processing time. Formal definitions of the proposed time-evolving co-author social network model are given below.
**Definition 6.2:** The co-author graph at time slice $t$ is represented as $G_t = (V_t, E_t, X_t)$ where $X_t$ represent the expertise vector that captures the knowledge of an agent and is defined in Equation (6.1).

$$X_t = \left\{ x^G_{v}^t \middle| x^G_{v}^t = [p_1, p_2, ..., p_r] \right\} \forall v \in V_t$$ \hspace{1cm} (6.1)

where $p_i = t f_i \times \log \left( \frac{N}{n_i} \right)$, $t f_i$ is the number of papers published by author $v$ at time $t$ on topic $i$, $N$ is the total number of papers at time $t$, $n_i$ is the number of papers on topic $i$ at time $t$ and $r$ is the total number of topics (Salton et al 1988).

**Definition 6.3:** The aggregated co-author network at time $t$ represented by $G^A_t$ is defined recursively as given in Equation (6.2).

$$G^A_t = \begin{cases} \phi & \text{if } t = 0 \\ \bigcup_{t=1}^{\beta} G^A_i & \text{if } t \geq 1 \end{cases}$$ \hspace{1cm} (6.2)

where the parameter $\beta$ specify the aging factor of the aggregated graph until time $t-1$. The value of $\beta$ is set to 0.5 in this study.

**Definition 6.4:** The threshold value to determine the expertise of an author at time $t$ is defined by a vector as $\theta_t = [h_1, h_2, ..., h_r]$ and $h_i$ is defined in Equation (6.3).

$$h_i = \max \left\{ p^r_i \left( x^A_{v}^t \right) \right\} - \epsilon, \forall v \in V^A_t$$ \hspace{1cm} (6.3)

where $h_i$ is the threshold value on the topic $i$, which is set to the maximum expertise value on topic $i$ reduced by a factor $\epsilon$. $p^r_i$ is the projection function that retrieves the $i^{th}$ value in the expertise vector associated with the author $v$, from the aggregated graph $G^A_t$. The author $v$ is considered to be an expert on topic $i$ if the value retrieved by the projection function is greater than or equal to $h_i$. 

**Definition 6.5:** To avoid flooding the query to all the neighbors of the authors, the query is selectively forwarded to few of their neighbors with expertise relevant to the query. The neighbor set $N(u, i)^{G_t^A}$ of any author $u$ at time $t$ for the topic $i$ includes $j$ best matching neighbors from the co-author set of $u$.

### 6.3.1 Learning and Expert Cache

The proposed approach captures the learning ability of an agent by including a learning factor $\alpha$. For empirical valuation the value of $\alpha$ is set to 0.3 in this study, however in reality it is determined subjectively by the querying agent in relevance to the answer it receives for its query on topic $i$. Additional knowledge gained by the agent is captured by incrementing its expertise vector value pertaining to topic $i$ by the *learning factor* $\alpha$. Also, answering agents are motivated to better respond to user queries by upgrading its expertise value for the query topic $i$ by a *motivational factor* $\gamma$. For simplicity reasons, the value of the motivational factor $\gamma$ is set equal to the learning factor $\alpha$ in this study. To appreciate other agents in the referral path their expertise vector value is increased by a factor, $\frac{\alpha}{2}$ as shown in Algorithm 6.2. Each agent is also associated with an expert cache that keeps track of the experts found during the query processing process. In future, if an agent receives a query on topic $i$ and experts on this topic is found in its expert cache, the agent sends the query directly to this expert that will reduce the depth of the referral graph.

### 6.4 Referral Based Expert Identification

The proposed Algorithm 6.1 locates an expert relevant to the given query. The algorithm takes as input the aggregated graph $G_t^A$ at a given time instance $t$, query generating agent (say $A_q$), query $W$ and experts pertaining to the query as output. The steps of the algorithm are explained below. Steps 1-4
initialize all the vertices in the aggregated graph except $A_q$ to be unmarked. This marking ensures an acyclic graph during query processing. $A_q$ maintains a queue $Q$ of agents that are found to be suitable to send the queries. Steps 5-7 of the algorithm initializes the queue $Q$ to the set of all neighbors, $N(A_q, i)^G_q$. A while loop encompasses steps 8-31 of the algorithm that finds an expert. Step 9 of the algorithm removes an element $v$ from the queue and checks whether it has been marked or not. If it is not marked, the steps 10-30 of the algorithm will be executed. In steps 11-12, the algorithm marks $v$ as marked and $A_q$ sends the query to $v$. The agent associated with node $v$: (1) may respond with an answer (provided it has got the required expertise), or (2) with a set of referrals or (3) may decide to refrain from further query processing.

---

**Algorithm 6.1 Expert Identification based on Referrals**

**Input:** Query generated by an agent $A_q$ at time slice $t$ on topic $i$.

**Output:** Returns an expert suitable to the query

1:   for (each author node $u \in G_t^A(V)$ ) do
2:       marked[$u$] = false
3:   end for
4:   marked[$A_q$] = true
5:   for (each author node $v \in N(A_q, i)^G_q$) do
6:       enqueue($Q$, $v$)
7:   end for
8:   while ($Q \neq \emptyset$) or (!timeout) do
9:       $v$ = dequeue($Q$)
10:      if (marked[$v$] = false) then
11:         marked[$v$] = true
12:            Agent $A_q$ sends query to $v$
13:            if ($v$ is an expert on topic $i$ and returns an answer) then
14:               $A_q$ evaluates the answer returned by $v$ and sets an appropriate value for the learning factor $\alpha$
15:               $P_t^r (x^i_{A_q}) = P_t^r (x^i_{A_q}) + \alpha$
Algorithm 6.1 (Continued)

16: \[ P_i^r(x_{v}^{G_i^A}) = P_i^r(x_{v}^{G_i^A}) + \alpha \]
17: update-edge-weight(v, depth(v))
18: \[ N(A,q_i)^{G_i^A} = N(A,q_i)^{G_i^A} \cup v \]
19: update the expert cache of author A_q to include v
20: return(v)
21: end if
22: else if (v returns referral) then
23: for (each author node \( u \in N(v, i)^{G_i^A} \)) do
24: \( v \) sends \( u \) as referral to \( A_q \) and \( A_q \) adds it to the queue, enqueue(Q,u)
25: end for
26: end if
27: else if (v does not return an answer or a referral within a specific time period) then
28: Ignore v
29: end if
30: end if
31: end while

Algorithm 6.2 update-edge-weight (v, int level)

Input: Expert Agent v for the query Q, depth of v
Output: Modified expertise values of the agents lying in the path between v and \( A_q \)

1: if (level\geq 0) and (v \neq A_q ) then
2: \( A_j = \text{parent}(v) \)
3: \[ P_i^r(x_{A_j}^{G_i^A}) = P_i^r(x_{A_j}^{G_i^A}) + \alpha \]
4: update-edge-weight(A_j,level-1)
5: end if

Steps 13-21 of the Algorithm 6.1 checks the first condition if v posses the required expertise to answer the query and aspires to respond with an answer to the query. Steps 22-26 of the algorithm checks the second condition, where an agent v may generate referrals if it does not possess the
required expertise to answer the query. The referrals are generated by selecting appropriate neighbors from the neighbor set of \( v \) and the selected neighbors of \( v \) will be sent as referrals to \( A_q \). \( A_q \) upon receiving the referrals, adds it to the queue \( Q \), and sends the query \( W \) to them later. Steps 27-29 of the algorithm check the third condition, where an agent \( v \) may decide to refrain from either responding with an answer or a referral. The Algorithm 6.1 is of linear time complexity as the while loop will be executed at most \( n \) times, where \( n \) is the total number of author nodes in the system.

6.5 EXPERIMENTAL ANALYSIS

The prototype of proposed Time Evolving Referral System (TERS) is implemented using Java 1.6. In this study, the social network of researchers in the domain of data mining has been constructed by collecting data from the proceedings of KDD (Knowledge Discovery and Data Mining) conference available at http://www.informatik.uni-trier.de/~ley/db/conf/kdd/, from the year 1994 to 2008. The conference data collected from the web is originally in the form of HTML files. The structures of HTML files were used to extract the titles of the paper and the authors’ information. Around 3200 authors and 1100 titles are identified in the conference data set.

This study reduces the titles to topics using a manually constructed taxonomy. The taxonomy consist of seventeen topics such as (Association Analysis, Clustering, Classification, Outlier Analysis, Learning, Prediction, Knowledge discovery, Query optimization, Pattern Analysis, Temporal and Time Series Mining, Spatial Mining, Concept description, Genetic Algorithm, Bayesian Classification, Decision tree, Text Mining, Web Mining). Each paper title is then reduced to one these topics. For example if keywords like frequent items, support, association rules confidence and interestingness
appear in the title, then the title will be reduced to the topic *association analysis*.

Each agent maintains an expertise model for its user, by associating expertise vector of size 17 (the number of topics in the taxonomy) with each author node. Based on the preset taxonomy this study could able to successfully reduce around 950 titles into topics. The number of papers under each topic are {Association Analysis(157), Clustering(101), Classification(138), Outlier analysis(15), Learning(51), Prediction(34), Knowledge discovery(171), Query optimization(19), Pattern analysis(54), Temporal & Time series mining(33), Spatial mining(4), Concept description(40), Genetic algorithm(21), Bayesian classification(20), Decision tree(13), Text mining(15), Web mining(56)}.

The fifteen years of conference data from 1994 to 2008 is divided into three time slices $G_1$, $G_2$, $G_3$ of five years each. The time evolving network is then constructed incrementally and while expanding the network the earlier time slice expertise of the authors are reduced by an aging factor $\beta$. The aging factor is included to give more weightage to the current expertise of the author.

Figure 6.3 show the number of experts found at each level for various combinations of the learning factor and expert cache. This graph reflects the six-degree separation phenomenon as referrals are very effective within length of six and decays after it.
Figure 6.3 Average Number of Experts Found with Learning and Expert Cache

6.5.1 Execution Time

Figure 6.4 shows the execution time taken to construct time evolving graph $G_t^A$ for each time slice and the static network covering the entire time period of the dataset. A substantial reduction in the execution time is obtained when the network is constructed in a time evolving manner rather than as a static network.

Figure 6.4 Execution Time in Time-Evolving Vs Static Co-Author Collaboration Network
6.5.2 Fan Out Factor

In the query processing process, when an agent does not have the required expertise to answer the query it generates referrals to its appropriate immediate neighbors. Fan out factor is the terminology used in this study to represent the number of neighbors to which the query is passed. It is a vital parameter as this helps the agent to have a focused search. Otherwise the query will be passed on to all the neighbors causing flooding disturbance. Experimentally, it is found that the optimal fan out factor is 4. Figure 6.5 shows the number of experts found for different fan out factors.

![Fan out Factor vs Number of Experts Found](image)

Figure 6.5 Number of Experts Found for Various Fan Out Factor Values

6.6 SUMMARY

Referral systems effectively capture the manner users in a social network help each other in finding trustworthy experts. Being decentralized it is more robust to failures and also effectively disseminates more knowledge across the system. In this chapter, a referral system has been proposed over a time evolving co-author network in the context of scientific collaborations. The experimental finding obtained provides a well defined strategy for an agent to selectively forward the query only to a few of its immediate neighbor rather than flooding it. The proposed system also has empirically proved the effectiveness of the expert cache in locating experts with less number of referrals.