Acta Informatica Pragensia
2(1), 2013, 1–17, DOI: 10.18267/j.aip.9
Online: aip.vse.cz

Section:
Peer-reviewed papers

A Transitive Recommendation System for Information Technology Communities

Waleed M. Al-Adrousy[1], Hesham A. Ali[2], Taher T. Hamza[1]

[1] Department of Computer Science, Faculty of Computers and Information
Mansoura University, Egypt
[2] Department of Computer Engineering and Systems, Faculty of Engineering
Mansoura University, Egypt

{waleed_m_m, h_arafat_ali}@mans.edu.eg, taher_hamza@yahoo.com

Abstract: Social networks have become a new trend for research among computer scientist around the world. Social network had an impact on users' way of life. One of social network usages is recommendation systems. The need of recommendation systems is arising when users try to know best choice for them in many items types (books, experts, locations, technologies, etc.). The problem is that a single person can't try all alternatives in all possibilities life goals to compare. Thus, a person has to use his friends' expertise to select better option in any item category. This process is the main idea of "Recommendation Systems". Recommendation systems usually depend on users-to-items ratings in a network (graph). Two main challenges for recommendation systems are accuracy of recommendation and computation size. The main objective of this paper is to introduce a suggested technique for transitive recommendation system based on users' collaborative ratings, and also to balance loading of computation. All this has to be applied on a special type of social network. Our work studied the transitivity usage in connections to get a relation (path) as a recommendation for nodes not directly connected. The target social network has eight types of nodes. So, there are techniques that are not suitable to this complex type of network. Those we can present a new support for recommending items of several types to users with several types. We believe that this functionality hasn't been fully provided elsewhere. We have suggested using single source shortest path algorithm combined with Map Reduce technique, and mathematically deduced that we have a speeding up of algorithm by 10% approximately. Our testing results shows an accuracy of 89% and false rejection of 99% compared to traditional algorithms with less configuration parameters and more steady count of recommendations.

Keywords: Social network, Recommender systems, Collaborative filtering, Parallelism, Web mining, Graph theory

1 INTRODUCTION

Social networks become one of human communications media. In social networks, users communicate to their friends using modern web 2.0 technologies. Each user can have many friends and in turn each friend can have many friends, so the transitive nature between users can lead to a *Friend-of-a-Friend* relationship (FoaF). Social networks environment encouraged the development of a trend called *"Recommendation Systems"*, this trend aims to recommend the best item of interest to a specific user based on his friends or his FoaF's recommendations – sometimes called "ratings". The recommendation systems themselves usually depend on social network and users' interactions with themselves and with other items (books, articles, code snippets, etc.). Usually those interactions can be stored in a log file or database. Sometimes, not all recommendations are trust worthy so a process of *Collaborative Filtering* (CF) is applied to remove anomalies of recommendations. According to [1], CF tries to identify users that have relevant interests and preferences by calculating similarities among user profiles. Recommender systems have several challenges [16]:

Accuracy of suggestion (recommendation), which measures how much similar the suggested items to user to his/her preferences.

1. Traceability, which means that system can present good reasons to user to know why the suggested items are selected. This should be presented in user-friendly way.
2. Timely calculation: whatever the techniques applied in recommendation systems, they should operate in a fast way for both users and server sakes.
3. Load balancing for huge calculations processes. This point can lead also to achieve the timely calculation goal (3rd point).

In this research, we are interested in 1st and 4th points (accuracy and load balancing). However, we add an extra dimension to the first point, which is to get the transitive relationships between nodes which are not directly connected. According to [16], there are several types of recommendation systems:

1. Rank-based recommendation: that's to calculate score for each item, and order items by it. This is similar to web ranking. Examples for these techniques are Tensor factorization and FolkRank [16].
2. Content-based recommendation: some semantic analysis is applied to content, meta-data or description of each item and then recommendation is calculated to measure how similar the item to each user profile.
3. User-based recommendation, that's to use other friends (or similar users) experts and get their suggestions.

Our work is focused on 3rd type of recommendation (user-based). This paper is organized as follows; some background and similar work are presented in section 2. The suggested technique is presented at section 3, with some mathematical analysis for algorithm speedup. In section 4, testing results and metrics are presented, and finally section 5 is the conclusion.

2 RELATED WORK

There are some common problems in researches about recommender systems [24]; prediction accuracy, testing opinions' subjectivity, recommended items suitability rather than being high ranked, effective preferences inference, trust formation of recommended items and usability of recommender systems interface layout. Those problems focus on the value of the recommended items rather than the performance of recommendation process itself. Recommender systems had many forms over the few past years. Some expert systems such as "Syskill & Webert" and Web Browser Intelligence (WBI) [25] applied intelligence to learn user preferences and patterns of recommendations and decision trees. This system allowed user to make symbolic ratings about items using words like cold or hot. The symbolic ratings were converted later by the system into numeric ratings [26]. Numeric rating aimed to handle the problem of difference in modeling nature between the human view and machine the view. However, this approach used explicit user likeness actions where user selects the rating for the item. Another technique in recommender systems was modeling of general interests was the analysis of user's navigation actions items pages. Also, another set of techniques is to use geographic location as a measure in recommendation systems for similarity, a good survey about this set is listed in [31].

Knowledge based filtering was discussed in the work of Bruke et al. [27]. Their work classified knowledge into three areas: the social knowledge about users' database, the individual knowledge about a particular user and content knowledge about the items in the network. They discussed some problems in knowledge based filtering such as the distribution of users to items. In that problem, few items gain the attention of users while some other good items may not discovered.

Some techniques of natural language processing and semantic analysis have been used in some researches such as the work of Santos and Boticario [28]. In that research a new concept of Semantic Educational Recommender Systems "SERS" was introduced to join e-learning with semantic analysis. In [29] a survey is introduced to explore the features of some social networks that include e-learning support. One of those discussed networks is "OpenStudy" which was a large social learning community.

Many researchers have studied social networks containing two types of nodes (bipartite graphs), such as users-to-items relationships. Items can be anything; products, videos or articles...etc. We shall present two families of related work: suggestion accuracy and Map Reduce application. Each of them is applied for network graphs. Sometimes those graphs are randomly generated with simulation rules while other researchers have used a real data set to test their work.

Firstly, for accuracy: According to [9, 10, 11, 12, 30], there are some researches to solve sparsity and cold-start problem. Ceylan and Birturk [19] have focused on semantic similarity calculation using three types of similarities; taxonomy (ontology analysis), relation and attributes. They made two predictors for ratings stored in a matrix. They suggested two models; SEMCBF (content based) and SEMCBCF (content and collaborative analysis). They could get a precision of about 63 and recall of range 92-93 approximately [19]. On the other hand, Purtell and his team [20] have developed a greedy algorithm for constructing social topologies from users' groups' communications data. Their work aimed to analyze emails and photo tags from users' profiles to build general information about social network. They applied their work on a collection of 1,995 email archives and 286,038 tagged photos

from Facebook website [20]. Some data mining techniques have been used for dimensionality reduction like Singular Value Decomposition (SVD) [13, 23]. By applying SVD, user-item rating matrix sparsity is reduced. Other researchers have used the semantic analysis of social content [14]. Another research technique was to use control theory and dynamics [15]. Papagelis and his team [1] suggested a technique in section 3 to use trust inference and user similarities estimation. They used Pearson co-relation to calculate recommendation for items. They also discussed the use of user similarities to enhance the recommendation.

Secondly, for Map Reduce with graphs: Ekanayake et al. have made [32] a good survey on applying Map Reduce for data intensive analysis. Morales and his team [21] have addressed the problem of social content matching using Map Reduce to handle very large data sets. Their algorithms (StackMR and GreedyMR) are tested on large data sets [21]. Both algorithms scale extremely well. However, StackMR requires only a poly-logarithmic number of Map Reduce steps, making it an attractive option for applications with very large data sets. Some other surveys were made in [17, 33, 34] about map reduce for parallel data processing.

Most of above work either focus on data mining side of computation, or the distributed computation mechanism. We combine both of them besides studying the multi-mode graph to get transitive recommendations for far away items.

2.1 CONTRIBUTION

In this paper we focus on a special case of graph of multi-type nodes, k-partite graph. To be specific we targeted an eight-partite network introduced in the following section. One importance of this eight-partite is the relation between the university sciences and real life career. Our work is focused on two major targets: parallelism of computation, and accuracy of suggestions. This paper suggests different technique for calculating recommendation rating queried by a user about an item. The technique of Map Reduce is suggested for load balancing. The algorithm of single source shortest paths is suggested to be used with combination with hierarchical clustering. Our work is tested in two parts; the first part is a mathematical part for Map Reduce effect, we concluded that we have a speeding up of algorithm by 10% approximately. The second part was a simulation testing for accuracy of suggestion. Our testing results for that part shows an accuracy of 89% and false rejection of 99% compared to traditional recommendation systems algorithms with less configuration parameters and more steady count of recommendations. We can claim that our work uniqueness is the combination of recommendation systems, Map Reduce, and academic learning co-relation with real life careers using eight-partite graph. An additional contribution is combining the previous goals with the ability to get transitive recommendations for items recommended by people not directly connected to starting user's friends. The nature of single source shortest path algorithms enabled that functionality.

3 SUGGESTED APPROACH

As we introduced in section 2, Papagelis and his team [1] suggested a technique in their paper (section 3) to use trust inference and user similarities estimation. This work is our start and we suggest using single source shortest pass algorithms to get a strongest relation of nodes in less time. We shall focus on Bellman-Ford [2, 3] algorithm applied on clustering results.

3.1 PROBLEM MODEL

Our main concern is based on a need for a broader range social networks, not just two-mode networks, but also multi-mode networks. In practice there is a relationships in information technology communities between many elements, we would focus on **eight types** of nodes: *developers, users, technologies, reusable components, version control repositories, students, educational courses and companies*. Figure 1 presents an overall relations graph between those elements. In Figure1 there's a symbol letter beside the name of each node (Element) to be used later in discussion, Each element has different view: **Developers set (*D*)** want to get best reusable components tested and documented and **technologies set (*T*)** for long term projects. Also, they may want to know which courses are best to get trained on to be up-to-date. Also, *Companies* want to hire the best developers, either for long or short term hiring. Or even to know which developer is good to use for outsourcing. Also, companies want to follow the trends of new technologies and have a decision when to migrate to a new technology.

Students set (*S*) want to learn new technologies and be up-to-date, and in the same time have a clear vision which academic educational courses are useful to their study. **Users set (*U*)** want to use or buy the best **Products set (*P*)** and have an advice from a friend which product is trusted. **Courses Curricula set (*C*)** need to be updated quickly to provide market needs. **Reusable Components and libraries set (*L*)** already have **version control repositories set (*R*)**. Those repositories have some useful data in their log, like frequency of updates, passed unit tests, quality assurance history and so on. Those data are important to estimate the quality of a product, but they aren't understood by regular users. So, if there's a simple understood measure of quality to users based upon quality nature, so more trust can be achieved.

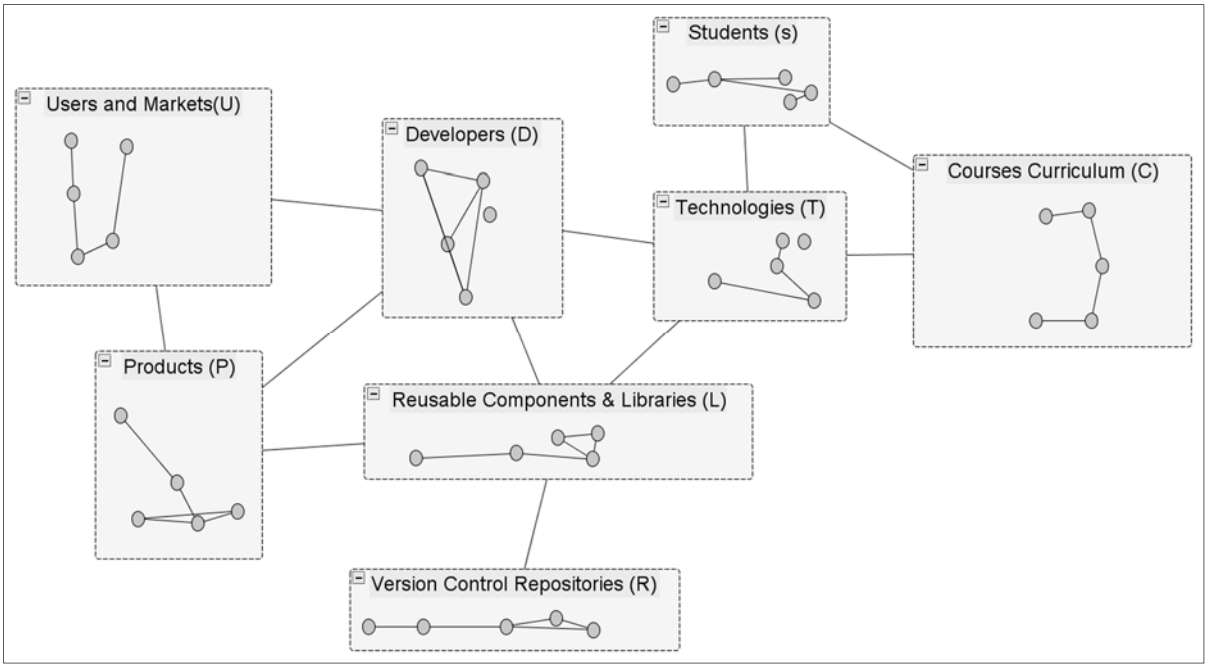

Fig. 1. Overall graph of relations nature between nodes.

According to relationships, there are three types of relationships: **Trust** relation (which is between humans and each other). **Usage** relation (which is between an item and its user). **Recommendation** relation (which is a higher degree of usage that has a degree of satisfaction). The social network itself

can be modeled as a large graph *G(V,E),* Where $v \in V, where\ V = \{S \cup U \cup P \cup C \cup L \cup R \cup T \cup D\}, \wedge \exists e \in E, where \mid e \mid \in \mathbb{Z}$. Usually, Social network edges values are limited to small set of integer values like set [1, 5] to express strength of relation between edge ends. Sometimes, some social networks express (Dislike) relation in negative values. There are some examples of applications in this large graph: Recommendation of products (P) to customers *U* (direct relation), recommendation of courses *C* to students *S* (direct relation), recommendation of courses *C* to students *S* though technologies (Indirect relation by filtering the direct relations between *C* and *S*), recommendation of Market trends *U* to students *S* though technologies *T*, products *P*, libraries *L* and so on. An important aspect is to get a relation between nodes not directly connected or not even from the same type or cluster. For example, to get a recommendation for students to learn new technologies based on the popularity of products developed by that technology.

3.2 PRELIMINARIES

3.2.1 BELLMAN-FORD ALGORITHM

Bellman–Ford runs in O(V·E) time [2], where V and E are the number of vertices and edges respectively. The Bellman-Ford algorithm determines the shortest path from the source *s* to each *v* in *V* for a graph *G = (V; E)* with real-valued weights, which may be negative. This point is important in case of expressing dislike operations in social network, which would result in negative edges between a user and an item. The algorithm assigns each vertex v in V its correct shortest path weight, provided there are no cycles with negative weights. The algorithm then returns true if and only if there are no negative weight cycles. The algorithm is described in Listing 1.

```
procedure Bellman Ford(list vertices, list edges, vertex source)
  // This implementation takes in a graph, represented as lists of vertices
  // and edges, and modifies the vertices so that their distance and
  // predecessor attributes store the shortest paths.

  // Step 1: initialize graph
  for each vertex v in vertices:
     if v is source then v.distance := 0
     else v.distance := infinity
     v.predecessor := null

  // Step 2: relax edges repeatedly
  for i from 1 to size(vertices)-1:
     for each edge uv in edges: // uv is the edge from u to v
        u := uv.source
        v := uv.destination
        if u.distance + uv.weight < v.distance:
           v.distance := u.distance + uv.weight
           v.predecessor := u

  // Step 3: check for negative-weight cycles
  for each edge uv in edges:
     u := uv.source
     v := uv.destination
```

```
    if u.distance + uv.weight < v.distance:
        error "Graph contains a negative-weight cycle"
```

List. 1. Bellman-Ford Algorithm [2, 4].

Bellman-Ford is preferred than another algorithm called Dijkstra shortest path [2, 4] for its complexity and running time.

3.2.2 SIMILARITY OF SAME-TYPE NODES

As presented in [1], similarity between users can be estimated using the following equation:

$$Similarity(u_x, u_y) = \frac{(\sum_{h=1}^{n}(r_{(ux,ih)} - \overline{r_{ux}})(r_{uy} - \overline{r_{uy}}))}{(\sqrt{(\sum_{h=1}^{n}(r_{(ux,ih)} - \overline{r_{uy}})^2)}\sqrt{\sum_{h=1}^{n}(r_{(uy,ih)} - \overline{r_{uy}})^2})}$$

From [1] the definition for this Pearson correlation uses the following symbols: u_x, u_y are two friend users. Items co-rated by both users are subset $I=\{i_x, x=1,2,....n\}$. $r_{ux,ih}$ rating of user u_x to item i_h. r_{ux}, r_{uy} are average ratings by users u_x, u_y.

3.2.3 MAP REDUCE

As explained in [22, 33, 36], Map Reduce is a parallel and distributed solution approach developed by Google for processing large data sets. Map Reduce has two key components. Map and Reduce. A map is a function which is used on a set of input values and calculates a set of key/value pairs. Reduce is a function which takes these results and applies another function to the result of the map function. The approach assumes that there are no dependencies between the input data. This makes it easy to paralyze the problem. Map Reduce depends on forking(Map) a computing over many processing units and aggregate their results (Reduce) to form final total result [8,17] as shown in figure 2:

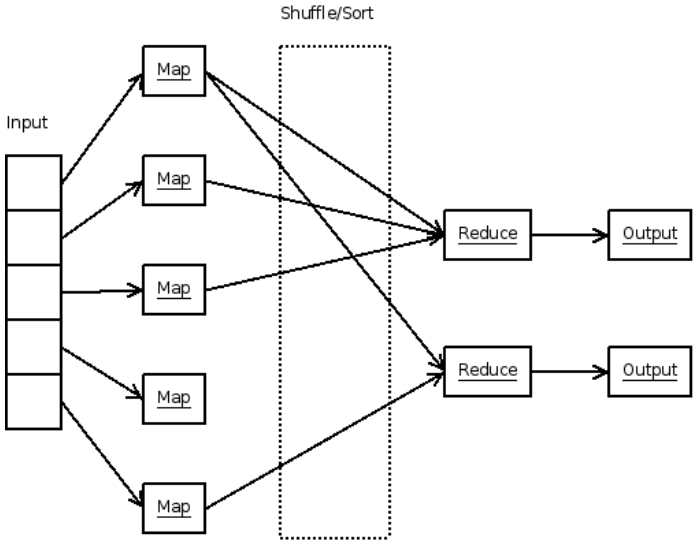

Fig. 2. Map Reduce Architecture according to [37].

This can reduce overhead on server and reduce calculations time.

3.3 SUGGESTED MAIN ALGORITHM

Instead of relaying only on either trusted friends' recommendations or similar users' recommendations, we suggest combining both of them in the following way:

Find Recommendations (Multi-Partite_network, partite_count, count of clusters):	**Complexity Estimation**
1. Consider each node a cluster initially	**O(*m*)**
*2. **Sort** all graph nodes based on hash value*	**O(m^2)**
*3. Find **proximity matrix** by getting all output edges weights for each node.*	**O(*m*)**
4. T = partite_count	**O(1)**
5. Initialize T of sub-proximity matrices, each matrix is for a set of nodes of same types.	**O(*m*)**
*6. **Parallel For** each sub-proximity matrix M* *Use Manhattan **distance** to cluster nodes inside same type sub-graph store the results as **connected components***	**O(m^2 +*u*)** *without parallelism*, **O(*m*+*u*)** *with parallelism.*
*7. **Merge** all connected components arrays into a single array.*	**O(m^2)**
*8. Create a **graph** of connected components,*	**O(1)**
*9. Let connected components graph edge weights = sum of all common edges between the nodes of those two components. **(edges grouping)***	**O(*m.u*)**
*10. **Remove weak relations** between connected components.*	**O(*u*)**
*11. Apply **Hierarchical clustering** to cluster connected components as blocks.*	**O(u.*log u*)**
*12. Apply **Bellman-ford shortest path** calculation on each cluster.*	**O(*u*.*e*)**
*13. Let **Recommendation set for each cluster = cluster neighbor of other types.***	O(*u*)

List. 2. Suggested Algorithm.

For edges in graphs, there is a problem; the nature of relationships between nodes differs. Some relationships that express "trust" can be expressed in range [-5,5]. However, dependency relations have broader range (like case of libraries and products). So, to unify our calculations we need a normalization pre-processing. For normalization, we suggest the following equation:

$$N_{(i,t)} = \frac{E_{(i,t)}}{(Max(E_t))}$$

Where *i* is the edge with index *i*, *t* the type of node, $E_{(i,t)}$ is the original edge weight for edge *i*, $N_{(i,t)}$ is the normalized edge *i* in type *t*, *Max*(E_t) is the greatest edge weight per same type *t*. in fact this is not a perfect normalization, but it's quick for computation. In the algorithm in listing 2, the 6th step is parallel, which enables the use of Map Reduce algorithm. In the following section, we shall present a mathematical analysis for the suggested algorithm to get an approximation of computation speedup. In second column in table in Listing 2, an estimation of each instruction complexity is presented to allow mathematical analysis in next sub-section.

3.4 PARALLELISM SPEEDUP APPROXIMATION

The speedup of calculation can be estimated by Dahlia's Law [18]:

$$speedup = \frac{1}{1 - p + \frac{p}{n}}$$

Where *p* is the percent of parallel code, and *n* is the number of processing nodes. In fact, this law is applied in multiprocessing systems originally. Of course, in distributed systems there should be a consideration for network overhead. However, the Dahlia's law is a good approximation for speedup. So in order to calculate the p factor we introduced a mathematical analysis for algorithm individual instructions in listing 2. First analysis for algorithm without parallelism: $Linear = 2 + 3 * O(m) + 2 * O(m^2) + O(u + m^2) + O(m) + o(m * u) + 2 * o(u) + O(u * \log(u)) + O(u.e)$

and since *u* and *e* are less than *m*, then:$Linear < 10 * O(m^2)$, and when algorithm has parallelism:

$$Parallelism = 2 + 3 * O(m) + 2 * O(m^2) + O(u + m) + O(m) + o(m * u) + 2 * o(u) + O(u * \log(u)) + O(u.e)$$

So, $Parallism < 9 * O(m^2)$, Then by dividing the parallelism by linear size, we can say that the parallelism had a decreased size percentage 0.9 approximately. So the percentage of parallel code (p) = 1-0.9 = 0.1 approximately. So we can apply Dahlia's law at p = 0.1 for any number of multiprocessing cores *n*.

4 TESTING AND RESULTS

4.1 SIMULATION ENVIRONMENT

Basically, we focused in this paper on testing the accuracy of transitive recommendation which is the second goal of algorithm. For the first goal (load balancing) we presented a mathematical analysis for it in sub-section (3.4). in that analysis we concluded that parallelism is effective by 10%

approximately. Running tests using JUNG[5] and LingPipe[6] java libraries for eight-partite graph for several random samples. The following figures show some samples of original testing networks (2 samples with different sizes), shown by figures 3 and 4.

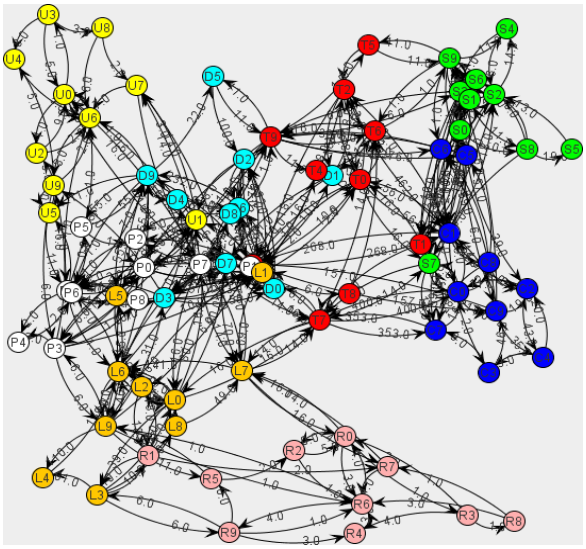

Fig. 3. Sample 1

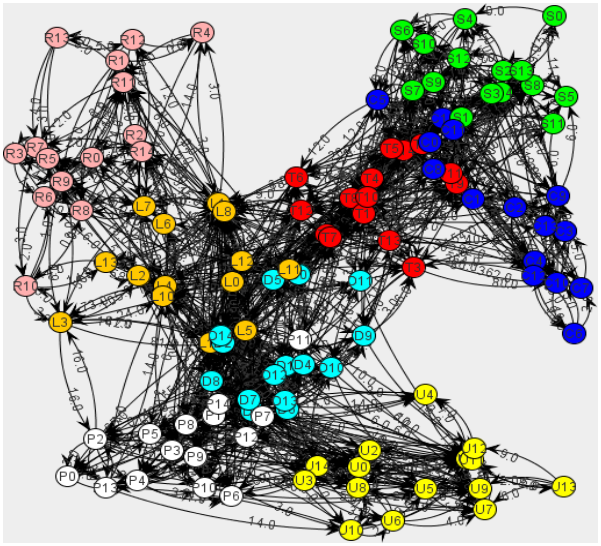

Fig. 4. Sample 2

In the our experiments, we randomly created graphs, containing eight types of nodes to represent eight types of objects in target study; students, courses, repositories, technologies, users, developers, products and reusable components. For simplicity, we've assumed that each type have equivalent number of items. Figure 3 shows a graph with 10 nodes for each type, while figure 4 shows another graph with 15 nodes per type. The edges in both graphs are also randomly created within a specified range for each type. The experiment method tried to compare traditional Collaborative Filtering techniques described in [1] with our suggested technique, and see how far our suggested algorithm could reach the same results using single source shortest path algorithm. After clustering both samples we get the following two connected components graphs shown in figures 5 and 6. The edges between clusters are the summation of outgoing edges weights in the children of each clustering to the other cluster children. After that a removal of weak edges is done. The value of threshold is fixed to an arbitrary value since it's beyond the scope of the current study.

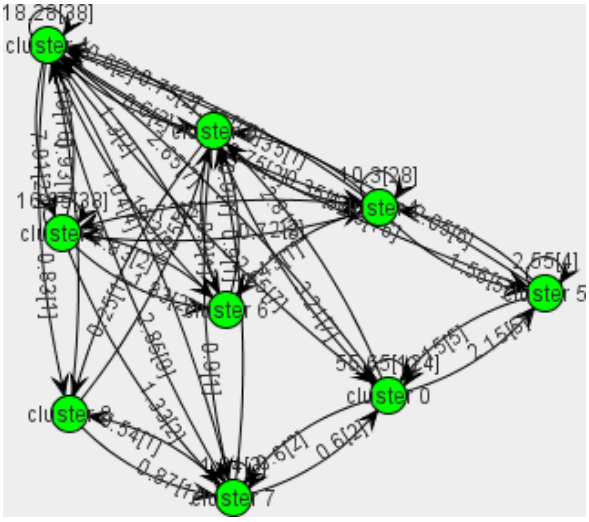

Fig. 5. Clustered Sample 1

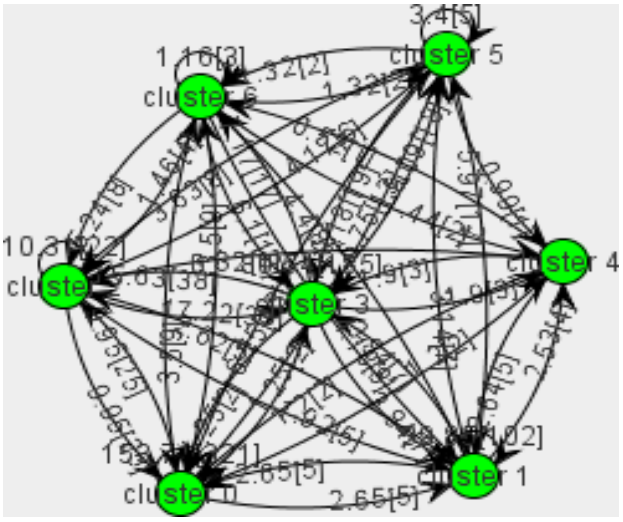

Fig. 6. Clustering sample 2

To evaluate algorithm we have applied several metrics discussed in the following sub-section to compare traditional recommender system algorithm with our suggested algorithm.

4.2 USED METRICS OVERVIEW

The used metrics for comparison is based on LingPipe library [6] precision evaluation. The *precision recall evaluation* is a matrix of counts of ***reference*** (goal) and ***response*** (tested) classifications. When a *tested* classification contains a *target* item then the value of the matrix is "*true*". Otherwise the "false" value is stored in matrix. To compare two classification results, the precision matrix is evaluated first, then iterating over all classes generated in matrix. The ideal target is to have all classifications sets common in those two matrix and no extra sets in any of them to 100% match. Table 1 presents all possible combinations between response and reference [6].

		Response matrix		**Reference Totals**
		True (found)	False (missing)	
Reference matrix	True (found)	*True Positive (TP)*	*False Negative (FN)*	*Positive Reference*
	False (missing)	*False Positive (FP)*	*True Negative (TN)*	*Negative Reference*
Response Totals		*Positive Response*	*Negative Response*	*total*

Tab. 1. The relation between a reference and response in precision matrix. [6]

From those cases we can get several statistics [6, 7, 35] presented in Listing 3.

1. $accuracy = \frac{correctresponsescount}{total}$
2. $recall = \frac{truepositivecases}{positivereference}$
3. $Precision = \frac{truepositive}{positiveresponse}$
4. $RejectionRecall = \frac{truenegative}{negativerefernce}$
5. $RejectionPrecision = \frac{truenegative}{negativeresponse}$
6. $referenceLikelihood = \frac{positivereference}{total}$
7. $responseLikelihood = \frac{positiveresponse}{total}$
8. $randomAccuracy = reflikehood * reslikehood + (1 - reflikehood) * (1 - reslikehood)$
9. $kappa = kappa(accuracy, randomaccuracy)$where $Kappa(p, e) = \frac{p-e}{1-e}$
10. $randomAccuracyUnbiased = AverageLikehood^2 + (1 - AverageLikehood)^2$

where:

$$AverageLikehood = \frac{referenceLikehood + responseLikehood}{2}$$

and $kappaUnbiased = Kappa(accuracy, randomaccuracyunibiased)$

List. 3. Summary of statistics used for evaluation.

In the following sub-section 4.3, a comparison is performed using the previous metrics.

5 EXPERIMENTS RESULTS

Consider the CF algorithm is reference and suggested clustered CF algorithm is response, we performed several experiments and we got those results.

Measure	AVERAGE
Accuracy	0،89
Rejection Recall	0،89
Rejection Precision	0،99
Random Accuracy	0،85
Random Accuracy Unbiased	0،85
kappa	0،25

Tab. 2. Testing results for comparing basic CF and Clustered CF.

The most important results are: accuracy from table [2] is 89%, and the rejection precision is 99%. So the, the suggested algorithm is moderate compared to original algorithm. However, Kappa agreement value is too low. We have an assumption that the too low kappa agreement is resulted due to extra transitive recommendation inferred by suggested algorithm. The discussion point is: are those extra

recommendations really suitable to querying start users? Well, this can't be covered in random simulation, and has to be tested on real users and get a real feedback to have some sort of supervised evaluation. Unfortunately, since our suggested eight-partite network is not covered a lot in previous researches, we didn't find suitable data set to test our algorithm on it. All available data sets are either same-type nodes with relationships or bi-partite networks (2 types of nodes only).

In order to explore in more detail the existence of extra recommendation by our algorithm compared to traditional collaborative filtering algorithm [1], we've made another experiment to compare count of suggested items in both algorithm in traditional recommendation shown in table 3 and Figure 7. Our experiment was designed to generate random network with fixed number of items per partite. After that, both traditional recommendation algorithm and our suggested algorithm are applied to the same network in each case. We've assumed that the traditional algorithm needs parameter *k*, as number of maximum recommended items per request. Of course, this parameter has many values, starting from 1 to *n* in case of network of items. So, in each network test case, we have tested many values for *k* in within that range. The values in table 3 are the result of dividing the value of suggested algorithm average count of recommendation over the traditional algorithm average count of recommendation.

$$ComparisonRatio = \frac{(Suggestedalgorithmaveragecountofrecommendation)}{(Traditionalalgorithmaveragecountofrecommendation)}$$

Items per partite	Maximum recommendation parameter k													
	1	3	5	7	9	11	13	15	17	19	29	**35**	**43**	**49**
10	10.00	10.00	5.00	5.00	3.30	-	-	-	-	-	-	-	-	-
12	12.00	12.00	6.00	4.00	4.00	4.00	-	-	-	-	-	-	-	-
14	14.00	14.00	7.00	4.66	4.66	3.50	3.50	-	-	-	-	-	-	-
16	16.00	16.00	8.00	5.30	5.30	4.00	4.00	3.20	-	-	-	-	-	-
18	18.00	18.00	9.00	6.00	6.00	4.50	4.50	4.50	3.60	-	-	-	-	-
20	10.00	10.00	5.00	3.30	3.30	2.50	2.50	2.00	2.00	1.60	-	-	-	-
30	10.00	10.00	5.00	3.30	3.30	2.50	2.50	2.00	2.00	1.60	1.40	-	-	-
50	10.00	10.00	5.00	3.30	3.30	2.50	2.50	2.00	2.00	1.60	1.25	**1.10**	**0.91**	**0.77**
100	10.00	10.00	5.00	3.30	3.30	2.50	2.50	2.00	2.00	2.00	2.40	**1.10**	**1.00**	**0.91**

Tab. 3. Matrix of Comparison Ratio.

So, when the comparison ratio is 2.0 for example, it means that our algorithm has produced twice as traditional algorithm (200%). In fact, our algorithm is steady in the number of items of recommendation; it produces almost the same count of recommendations and doesn't need extra parameter of *k*. In fact it's the same value in column at *k*=1 in table 3. We assume that is easier for users who don't want to enter configuration values like *k*. the dashed values in table 3 means that *k* value is not applicable in that case. An interesting result has been noticed, that as more convergent the value of *k* to the value of count of items per partite, the more similar count of items produced by the two algorithms and at some point our algorithm produces less results than the traditional algorithm. To judge which is more correct, we need a real sample data set with a real feedback of users to get some

sort of supervised learning. In figure 7, a curve is made for each sample with fixed count of items per partite. The curve values themselves are the comparison ratio values defined in table 3.

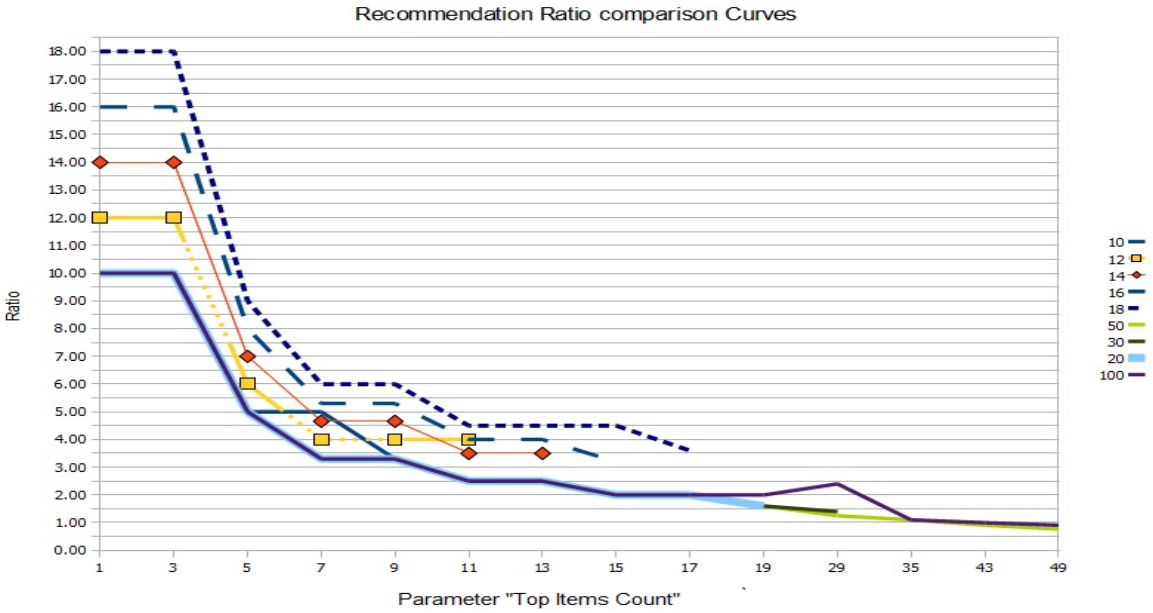

Fig. 7. Recommendation Ratio Comparison Graph.

6 CONCLUSION

The suggested algorithm is designed for two main goals: load balancing and accuracy. For first goal (load balancing), it could use parallel loops using Map Reduce technique to get a reduction of computation size estimated mathematically as 10%. for second goal (accuracy of recommendation), it achieves 89% accuracy and 99% rejection precision with less configuration parameters and more steady count of recommendations. The power of the suggested algorithm is in finding association between k-partite graphs compared to other algorithms which may ignore the nature of k-partite. The use of Map/reduce is proposed to reduce overhead on server and reduce computation time. The practical application of suggested algorithm in this paper can achieve the mutual use of distributed systems algorithms, web mining and graph theory techniques. However, testing results is based on simulation. We hope that we can apply this in real life and test the suggested algorithm and architecture when dealing with target eight-partite graph in technology community. Our results show that the suggested algorithm is a good start but needs more enhancements in the quality of recommendation. In fact, we have a problem in randomly created samples, that can't confirm or deny that the suggested algorithm recommendations are really suitable to users, or has some shortage in some case. Some real test cases and datasets are needed.

7 REFERENCES

[1] PAPAGELIS, M., PLEXOUSAKIS, D., KUTSURAS, T. Alleviating the Sparsity Problem of Collaborative Filtering Using Trust Inferences. *Trust Management,* 2005, pp. 224-239.

[2] CORMEN, T. H., LEISERSO, CH. E., RIVEST, R. L., STEIN, C. *Introduction to Algorithms*, Third Edition. The MIT Press, 2009.

[3] MEGHANATHAN, N. Survey of Topology-based Multicast Routing Protocols for Mobile Ad hoc Networks, *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 3, no. 2, pp. 124–137, 2011.

[4] JENSEN, J. B., GUTIN, G. The Bellman-Ford-Moore algorithm. Section 2.3.4 in *Digraphs: theory, algorithms and applications*. Springer, 2009.

[5] Jung Library [online]. [cit. 2013-05-21]. URL: http://jung.sf.net/

[6] LingPipe online documentation [online]. [cit. 2013-05-21]. URL: http://alias-i.com/lingpipe-3.9.3/docs/api/com/aliasi/classify/PrecisionRecallEvaluation.html

[7] CHOI, S. S., Cha, S. H., TAPPERT, C. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics* 8.1, pp. 43-48, 2010.

[8] JI, CH., BUYYA, R. Mapreduce programming model for. net-based cloud computing. *Euro-Par 2009 Parallel Processing*. Springer Berlin Heidelberg, pp. 417-428, 2009.

[9] BREESE, J. S., HECKERMAN, D., EMPIRICAL, C. K. Analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, San Francisco, Morgan Kaufmann, pp. 43–52,1998.

[10] MELVILLE, P., MOONEY, R. J., NAGARAJAN, R. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, (AAAI/IAAI-02), AAAI Press, Menlo Parc, CA, USA, pp. 187–192, 2002.

[11] XU, B., BU, J., CHEN, C., CAI, D. An exploration of improving collaborative recommender systems via user-item subgroups, In *Proceedings of the 21st international conference on World Wide Web - WWW* '12, April 16–20, 2012, Lyon, France, p. 21-30, 2012.

[12] CEYLAN, U., BIRTURK, A. Combining Feature Weighting and Semantic Similarity Measure for a Hybrid Movie Recommender System. *The 5th SNA-KDD Workshop '11* (SNA-KDD'11), August 21, 2011, San Diego CA USA.

[13] SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., RIEDL, J. T. Application of dimensionality reduction in recommender systems: A case study. In *WebKDD Workshop at the ACM SIGKKD*, 2000.

[14] JANE, J. Y. L., HSU, Y. J. LEE, T. Y. News Feed Filtering with Explanation Using Textual Concepts and Social Contacts. *The 5th SNA-KDD Workshop '11* (SNA-KDD'11), San Diego CA USA, August 21, 2011.

[15] JAMBOR, T., WANG, J., LATHIA, N. Using Control Theory for Stable and Efficient Recommender Systems. In *Proceedings of International World Wide Web Conference Committee* (IW3C2), Lyon, France .April 16–20, 2012.

[16] MARINHO, L. B., NANOPOULOS, A., THIEME, L. S. Social Tagging Recommender Systems, chapter in *Recommender Systems Handbook*, pp. 615–644, 2011.

[17] LIN, J., SCHATZ, M. Design patterns for efficient graph algorithms in *MapReduce. Proceedings of the Eighth Workshop on Mining and Learning with Graphs.* ACM, pp.78-85, 2010.

[18] DONIS, M. *Parallel Programming with Microsoft® Visual Studio® 2010 Step by Step*. Microsoft Press, 2011.

[19] CELYAN, U., BIRTURK, A. Combining Feature Weighting and Semantic Similarity Measures for Hybrid Movie Recommender System. *The 5th SNA-KDD workshop '11*, San Diego, CA USA, August 2011.

[20] PURTELL, T. J., MACLEAN, D., KEAT, S. An Algorithm and Analysis of Social Topologies from Email and Photo Tags. *The 5th SNA-KDD workshop '11*, San Diego, CA USA, August 2011.

[21] MORALESM, G. D., GIONIS, A., SOZIO, M. Social Content Matching in Map Reduce. *The 37th international conference on very large databaese*, Seattle, Washington. August-September 2011.

[22] VOGEL, Lars. MapReduce Introduction [online]. [cit. 2013-05-29]. URL: http://www.vogella.com/articles/MapReduce/article.html

[23] ZHANG, Y., ZHOU, J., CHENG, J. Preference-Based Top-K Influential Nodes Mining in Social Networks, in *2011I EEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, 2011, pp. 1512–1518.

[24] PU, P., CHEN, L., HU, R. A user-centric evaluation framework for recommender systems. Proceedings of the *ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI)*, Barcelona, Spain, Sep 30, 2010, i, pp. 14–21.

[25] GOMAH, A., ABDEL-RAHMAN, S., BADR, A., FARAG, I. An Auto-Recommender Based Intelligent E Learning System. *International Journal of Computer Science and Network Security*, 2011, 11, (1), pp. 67-70.

[26] LOPS, P., DE GEMMIS, M., SEMERARO, G.: Content-based Recommender Systems: State of the Art and Trends, chapter in: *Recommender Systems Handbook* (Springer Science and Business Media, LLC), 2011, pp. 73–105.

[27] BURKE, R., FELFERNIG, A., GKER, M.: Recommender systems: An overview. *AI Magazine*, Association for the Advancement of Artificial Intelligence, 32, (3), pp. 13–18, 2011.

[28] SANTOS, O., BOTICARIO, J. Requirements for Semantic Educational Recommender Systems in Formal E-Learning Scenarios. *Open access Algorithms*, 2011, 4, (2), pp. 131-154.

[29] RAM, A., AI, H., RAM, P., SAHAY, S. Open Social Learning Communities. *International Conference on Web Intelligence, Mining and Semantics*, WIMS-11, Sogndal, Norway, May 2011, Article No. 2.

[30] P. LOPS, DE GEMMIS, M., SEMERARO, G. Content-based Recommender Systems : State of the Art and Trends, *Springer Science and Business Media*, LLC, 2011, pp. 73–105.

[31] DESROSIERS, C., KARYPIS, G. A comprehensive survey of neighborhood-based recommendation methods, chapter in *Recommender Systems Handbook*, 2011.

[32] EKANAYAKE, J., PALLICKARA, S., FOX, G. Mapreduce for data intensive scientific analyses. eScience, 2008. *eScience'08*. IEEE Fourth International Conference on. IEEE, 2008.

[33] LEE, K. H., et al. Parallel data processing with MapReduce: a survey. *ACM SIGMOD Record* 40.4 (2012): pp.11-20, 2012.

[34] DELIP, R., YAROWSKY, D. Ranking and semi-supervised classification on large scale graphs using map-reduce. *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics, 2009.

[35] MANNING, C. D., RAGHAVAN, P., SCHÜTZE, H. *Introduction to information retrieval*. Vol. 1. Cambridge: Cambridge University Press, 2008.

[36] DEAN, J., SANJAY, G. MapReduce: simplified data processing on large clusters. *Communications of the ACM 51.1*, pp.107-113, 2008.

[37] Werneck Paiva. [online]. [cit. 2013-04-29]. URL: http://blog.werneckpaiva.com.br/2011/08/como-funciona-o-map-reduce-usado-pelo-google/