computers

Article

Modeling Bimodal Social Networks Subject to the Recommendation with the Cold Start User-Item Model †

Robert Albert Kłopotek

Faculty of Mathematics and Natural Sciences, School of Exact Sciences, Cardinal Stefan Wyszyński University in Warsaw, 01-938 Warszawa, Poland
† Conference on Information and Software Technologies (ICIST 2019)

Received: 3 January 2020; Accepted: 6 February 2020; Published: 12 February 2020

Abstract: This paper describes the modeling of social networks subject to a recommendation. The Cold Start User-Item Model (CSUIM) of a bipartite graph is considered, which simulates bipartite graph growth based on several parameters. An algorithm is proposed to compute parameters of this model with desired properties. The primary desired property is that the generated graph has similar graph metrics. The next is a change in our graph growth process due to recommendations. The meaning of CSUI model parameters in the recommendation process is described. We make several simulations generating networks from the CSUI model to verify theoretical properties. Also, proposed methods are tested on real-life networks. We prove that the CSUIM model of bipartite graphs is very flexible and can be applied to many different problems. We also show that the parameters of this model can be easily obtained from an unknown bipartite graph.

Keywords: social network analysis; recommendation; network graphs; bipartite graphs; bipartite graph model; graph growth simulation

1. Introduction

A social network often means the social structure between actors, which are generally individuals or individual organizations. It shows relationships of various types, ranging from random acquaintances to the close relationship, or to object flows (e.g., information, goods, money, signals, intermediates in the production cycle) between members of the community [1].

Social network analysis (SNA) is focused on mapping and measuring relationships and information flows between people, their groups, organizations, or other entities in transforming information and/or knowledge. SNA attempts to make a prediction on the basis of the characteristics of the network as a whole entity, the properties of individual nodes based on network structure, and so forth. The subject of the research can be a complete social network, or parts of it can be related to a specific node.

Nowadays, graphs are used to model various interesting real-world phenomena. Much interest of researchers has been attracted by social networks in which one can distinguish between two types of objects, such as users and items, and where relationships only between a user and an item are of interest. They can be modeled via bipartite graphs, which are graphs in which edges exist only between two disjoint subsets of vertices. For example, in the case of customer data, there are two modalities: users and products. There are no edges between users in the user set, and there are no edges between products in the product set. An edge between a user and a product means that the user bought this product. Such graphs can be utilized to recommend some products to users. Another example is an Internet forum. In this case, there are two modalities: forum users and forum threads in which they write posts. There would be an edge between a forum user and a thread if the user wrote a post on it. One can recommend some interesting forum threads for the user. One may also seek for intermediate states of a dynamic network that have not been observed.

Both the actual graph structure and the graph dynamics and its development in time are essential. Such growth models are vital in SNA for a number of goals. The first one is to test which microoperations happening in the network may lead to the macrostructures that one can observe. The second reason is that one wants to develop and test various social network algorithms, such as recommendation algorithms based on social networks, but the available social networks are not numerous, and the threat of overfitting is serious. Therefore, one needs synthetic networks which are similar to real ones. The third reason is that one may want to perform some kind of what-if analysis on social networks without experimenting with real people. Many more reasons can be found. For the above-mentioned purposes, on the one hand, one needs growth models that are sufficiently similar to real-world phenomena, and on the other hand, one also requires a method of extracting model parameters from the actual real network in order to generate similar ones.

Over the last decade, a number of growth models for bipartite graphs have been proposed [2–4]. Unfortunately, these bipartite graph generators have had some limitations. The bipartite graphs were created with limited reproduction of real-life graph properties, and two graph structures were also created, which complicates the models a lot.

In this paper, the graph generator proposed by Chojnacki [5] is considered, which can be viewed as a graph growth model with seven parameters. In [5], it has been demonstrated that the model qualitatively reflects properties of real-life bipartite graphs quite well. Therefore, it may be, and has been used for qualitative studies of various phenomena. Chojnacki's model touches on a very important problem of "cold start" in the recommendation of products to users, and vice versa. The "cold start" problem concerns the recommendation of products to a new user from whom one has no information in the system. The same occurs when one has a new product that one has no information about and wants to recommend it to users in the system. Thus, from here on, this model will be called the *Cold Start User-Item Model (CSUIM)*.

Our long-term goal is to investigate CSUIM's usefulness for quantitative analysis. Assuming that the real-world graph follows the growth paradigm described by CSUIM, this means that we want to identify the growth parameters of the graph so that one can, for example, investigate the growth of this graph in the past or in the future.

Regrettably, no results are known so far for computing or estimating model parameters from the real-world data for CSUIM. The current paper is intended to close this gap, and attempts to estimate to what extent the model parameters can be properly recovered from the graph in order to later on answer the question of how the application of recommendations onto the participants of a social network may change the social graph growth process. Therefore, artificial graphs generated from the Chojnacki model are studied in this paper, and the model recovery method, proposed in this paper, is applied to them.

In this paper, the first stage that is considered is methods of reconstructing generator models from the graph at some stage of development. A method to capture the parameters from the actual graph is proposed, and the similarity of metrics between the original graph and the one obtained from the model is verified.

Chojnacki used his model for other purposes. He created a benchmark framework for recommendation systems. His model examines how the recommendation system would behave, and was applied for the generation of different graphs.

The paper is structured as follows: Section 2 presents attempts to describe real-world phenomena of uni-modal and bi-modal social networks available in the literature. In Section 2.1, uni-modal graph models are described, and ideas used in the bipartite graph model are outlined in Section 2.2. In Section 3, the Chojnacki generator is mentioned briefly. Section 5 presents theoretical node degree distribution models. The proposed approach to parameter estimation is described in Section 6, and some linear dependencies for parameter estimation are investigated in Sections 7 and 8. In Section 9, a method for parameter computation from a graph is proposed. In Section 10, experimental results on parameter recovery and model quality are presented. Section 11 contains some concluding remarks.

2. Related Work

Much research efforts have been devoted to the qualitative description of the real-world phenomena of uni-modal social networks. Barabási [6] coped with the impact of the removal of a few super-connected nodes, or hubs. Albert and Barábasi [7] present a statistical approach to modeling random graphs, small-worlds, and scale-free networks, evolving networks, and the interplay between topology and the network's robustness against failures and attacks.

Lin et al. [8] used network history to predict communities in the current state, exploiting node degrees, modularity (as defined by Newman et al. [9]), and their own soft modularity.

Leskovec et al. [10] characterized the statistical and structural properties of communities as a function of network size and conductance.

Leskovec et al. [11,12] investigated the phenomenon of real graphs densifying over time, and shrinking of the average distance between nodes. They attempted to explain the phenomenon by models of "Community Guided Attachment" (CGA) and a more complex "Forest Fire Model" (FFM).

While there are many publications concerning uni-modal social network growth models, bimodal ones are far more rarely investigated, though as [13] shows, they are important for product recommendation and rating prediction, or as [14] (sec. 6.4.4.) reports, they may be used for investigation of models of scientific paper co-authorship.

Publications like [15] or [16] propose models for new edge prediction.

The paper of Lavia et al. [17] is the most similar in spirit to our work. In their paper, the Netflix competition database was considered, and an explanation for the hardness of prediction was made. The authors there proposed a growth model of an item rating network based on a mixture of preferential and uniform attachment that reproduces the asymptotic degree distribution, but also agrees with the Netflix data in several time-dependent topological properties.

Our research differs from this in that we are considering a much more complex model, where both items and users can perform the edge attachment, and a bouncing mechanism for modeling the impact of local recommendation is included.

2.1. Graph Models for Unimodal Networks

In this section, the most popular graph generators are presented, also called graph models.

First, let us recall an important measure of graphs, which is frequently used when evaluating the quality of various graph models of social networks.

Many empirical graphs are well-modeled by small-world networks (see [18]). For example, social networks, the connectivity of the Internet, wikis such as Wikipedia, and gene networks—all of them exhibit small-world network characteristics.

Therefore, in the literature, a couple of measures have been proposed to determine whether a graph is a small-world network. The most popular of them is the so-called local clustering coefficient (LCC), and for a vertex, i it is defined as:

$$LCC(i) = \frac{|(a,b) \in E : (a,i) \in E \wedge (b,i) \in E|}{k_i(k_i-1)/2}, \tag{1}$$

where E is the set of all edges, V is the set of all vertices, $a, b \in V$ are vertices, and k_i is the degree of vertex i. The degree of vertex i is the number of edges incident to this vertex. For the whole graph G, the clustering coefficient is just $LCC(G) = \sum_{i \in V} \frac{LCC(i)}{|V|}$.

The Erdös-Réni model (see [19]) is defined by two parameters: the number of nodes, n, and the probability that there exists an edge between nodes, p. This mechanism of node connection is called uniform attachment (UA). In this model, the node degree distribution follows the exponential distribution and local clustering coefficient $LCC \sim n^{-1}$. The Bárabasi–Albert model (see [7]) uses a "preferential attachment" (PA) mechanism for creating a connection between nodes. A graph is initialized with a connected graph with m_0 nodes. In the following steps, each node is connected to m existing nodes in such a way that the probability of connection is proportional to the number of links that the existing nodes already have. This method of connecting nodes causes that node degree distribution to follow a power-law distribution $P(k) \sim k^{-3}$ and $LCC \sim n^{-3/4}$. Liu's model (see [20]) was one of the first attempts at combining the uniform attachment and preferential attachment mechanisms. The authors proposed a parameter ϱ of intensification, mediating between UA and PA.

The models mentioned previously have serious drawbacks—the LCC value does not depend on graph parameters. More flexible models were proposed by Vázquez (see [21]), and independently by White (see [22]), where LCC may be modified by changing graph parameters. In the Vázquez model, the idea is based on random walks (called also surfing) and a recursive search for generating networks. In the random walk model, the walk starts at a random node, follows links, and for each visited node, with some probability, an edge is created between the visited node and the new node. It can be shown that such a model generates graphs with a power-law degree distribution with an exponent greater than or equal to 2 (see [23]).

2.2. Graph Models for Bimodal Networks

A bimodal network is understood as a network connecting two varieties of objects (like authors and their papers, employees and their firms, tourists and museums, etc.) [24]. Other names for such a network are a bipartite, 2-partite, or 2-mode network. These networks can be modeled by bipartite graphs. A bipartite graph has the form $G = (U \cup V, E)$, where $U \cap V = \emptyset$, and $E \subseteq U \times V$. That is, vertices of the bipartite graph can be divided into two disjointed sets, U and V, such that every edge connects a vertex in U to another one in V; that is, U and V are independent sets. These sets represent, for example, customers and products. If a customer u_i buys a product v_j, there is an edge between vertex u_i and v_j. Thus, there are no edges between customers and between items—they cannot buy each other. In the case of an Internet forum, one could also have two modalities: one for users and the other for threads the users participate in. Many other kinds of bipartite networks occur in real life [25].

Although bimodal graphs are a specific subclass of uni-modal graphs, models mentioned in Section 2.1 are not appropriate when one wants to model bipartite graphs of bimodal social networks. In the bipartite graph for all vertices a, b in the same modality set, one does not have any edges between them, so one always gets $LCC = 0$. This means there is a severe problem when studying bipartite graphs, because, on the one hand, one does not have any means of looking at the fundamental small-world phenomena, and on the other hand, it is an obstacle in adopting traditional graph generators to the case of bipartite ones. Therefore, in [5], another suitable metric for clustering tendency was proposed—the bipartite local clustering coefficient (BLCC):

$$BLCC(u) = 1 - \frac{|N_2(u)|}{\sum_{v \in N_1(u)} (k_v - 1)}. \tag{2}$$

W is the set of all vertices, $N_s(n)$—the set of neighbors of vertex $n \in W$, which are $s \geq 1$ steps away. In other words, $N_s(n) = \{a \in W : K(n, a) = s\}$, where $K(i, j)$ is a minimal distance (number of edges) between vertices i and j. In [5], it is shown that the graph metric LCC and $BLCC$ are similar in classical graphs.

Typically, in a social network, a small number of vertices have many direct neighbors, and a large number of vertices have a small number of direct neighbors. Social networks contain clusters with a high density of connections. This network property is called transitivity, and says that if nodes a and b have a common neighbor, then this influences the probability of the existence of an edge between a and b. The critical difference between unimodal and bimodal networks led to the development of separate models for the bimodal case. Let us mention a few.

There are two main approaches to model bipartite graphs: the iterative growth method and the configuration method. The iterative growth method is better for the recommendation process, as shown in [5]. It simulates the growth of a network. In the configuration method, one gives their estimated general description of the network, and from this, one constructs the final state of the network. The description usually contains: the number of nodes in each modality, the probability density function (PDF) of the nodes' degree, and the number of edges. Then, one creates nodes in each modality without edges. One creates edges from sampling endpoints of the edge from the node degree probability density function (PDF).

In [4] Guillaume and Latapy presented a method of transforming the bipartite graph into a classical network (uni-modal) and of reversing this transformation. Unfortunately, the reverse transformation is not unique, and moreover, the retrieval of the bipartite structure is computationally hard. The authors pointed out that the computation of the largest clique containing a given link may be very expensive (it is NP-complete). Birmele [2] builds a bipartite graph model from existing uni-modal graph models using retrieval of the bipartite structure from classical graphs. In [3], Zheleva et al. analyzed the evolution of groups in an affiliation network. The affiliation network has two modalities: users, and groups to which users belong. In the co-evolution model, groups can disappear and merge. This model is not appropriate in the case of recommendation items for users—items do not merge. In [26], Lattanzi and Sivakumar proposed a different model of the affiliation network as the bipartite graph. Their model for the evolving affiliation network and the consequent social network incorporates elements of preferential attachment and edge copying. They analyze the most basic folding rule, namely, replacing each society node in the affiliation network by a complete graph on its actors in the folded graph. The drawback of their models is that given a social network (or another large graph), it is not at all clear how one can test the hypothesis that it was formed by the folding of an affiliation network. The general problem of solving, given a graph G on a set Q of vertices, whether it was obtained by folding an affiliation network on vertex sets Q and U, where $|U| = O(|Q|)$, is NP-Complete.

All previously mentioned generators have an iterative growth mechanism. The common limitation of those generators is that they generate bipartite graphs with the power-law or uniform distribution of vertex degrees. Additionally, none of these models contains a parameter which controls the transitivity property. The previous approaches also have a significant drawback: configuration methods and methods based on retrieval of a bipartite structure decrease the bipartite local clustering coefficient (BLCC) compared to iterative methods. This means that models derived by those methods fit the real-world structures worse than those estimated by iterative methods. Moreover, in the pessimistic case, one deals with the NP-complete problem, so some approximations are needed. As these models suffered from various drawbacks, in [5], another model was proposed, which is characterized in Section 3 and which is the subject of our current investigation.

2.3. Recommender Systems for Bimodal Networks

Bimodal networks appear as a natural setting for a recommendation system, where objects of one modality are recommended for the objects of the other modality. We have already mentioned the works [5,26], and another addressing modeling for recommendations under these settings—however, there are many more. Ahmedi et al. (see [24]) derived recommendations from a network associating tourists with points of interest, based on the vertex and edge labeling combined with some ranking or centrality function. He et al. (see [27]) proposed to predict item popularity and to recommend items to users based on a specific version of a PageRank technique (eigenvectors of a special connectivity matrix). User preferences are expressed as weights. Shi et al. (see [28]) used for recommendation a combination of content-based and collaborative filtering, while a method of combining both recommendations was developed via learning the weights of both components from previous prediction accuracy. Cheng et al. (see [29]) exploited a matrix factorization model based on reviews and preferences. Ozsoy (see [30]) proposed to use the word2vec technique, originally developed for seeking words occurring in similar contexts. This approach replaces words with users and items, creating a kind of word2vec representation of the item–user graph. Recommendations are based on the similarity of objects in this representation. Vasile et al. (see [31]) proposed, based on the same word2vec technology, recommendations of items (products) based on their context (other products), as well as some textual information (content-based support). Kang and Yu (see [32]) developed a soft-constraint-based online LDA algorithm for community recommendation. It also accommodates a technique used for document processing to the collaborative filtering setting. A user is represented as a "document", being a probability distribution over latent topics, and each topic is represented as a probability distribution over communities. The number of users' posts within each community forms the foundation for estimation of latent topics, whereby an online LDA algorithm is applied for this purpose. Communities are recommended based on the conditional distribution of a community against the user "document". Liu et al. (see [33]) developed a recommendation method enriching online LDAs with probabilistic matrix factorization. Other application cases are reviewed in [34].

The current paper proposes a framework that differs from the just-mentioned approaches. They attempt to make recommendations taking into account the current state of the network. In the approach presented in this paper, the history of the network is modeled—that is, the predictions are related to the evolution of the network, and not to a suggested recommendation to a particular user at a given snapshot.

3. CSUIM Bipartite Graph Generator

The bimodal graph generator presented in [5] is more flexible than graph generators mentioned in Section 2.2, though it cannot generate a disconnected graph with desired properties. Its advantage is the capability to create graphs with a broader range of clustering behavior via the so-called bouncing mechanism. The bouncing mechanism is an adaptation of a surfing mechanism in classical graphs (see [21]). The bouncing mechanism is used only to the edges, which were created according to the preferential attachment.

In the CSUIM, we consider a graph with the set of vertices $W = U \cup V$, $U \cap V = \emptyset$, where the set U is called "users" and set V is called "items". We consider both the uniform attachment, where incoming nodes form links to existing nodes selected uniformly at random, and the preferential attachment, when probabilities are assigned proportional to the degrees of the existing nodes (see [35]).

The generator has seven parameters:

1. m—the initial number of edges, where the initial number of vertices is $2m$
2. δ—the probability that a new vertex v added to a graph in the iteration t is a user $v \in U$, so $1 - \delta$ means the probability that the new vertex v is an item $v \in V$
3. d_u—the number of edges added from the vertex of user type in one iteration (number of items bought by a single new user),
4. d_v—the number of edges added from the vertex of item type in one iteration (number of users that bought the same new item)
5. α—the probability of *item* preferential attachment, $1 - \alpha$—the probability of *item* uniform attachment
6. β—the probability of *user* preferential attachment, $1 - \beta$—the probability of *user* uniform attachment
7. γ—the fraction of edges attached in a preferential way, which were created using the bouncing mechanism

The *Cold Start User-Item Model (CSUIM)* creates a node in the set of users with probability δ and $1 - \delta$ in the set of items. The newly created node is connected with nodes of the opposite modality. If the node is of user type, it will be connected with d_u items, and if it is of item type, then it will be connected with d_v nodes of user type. To find the node to which the newly added node will be connected, we use two mechanisms: the "uniform attachment" (UA) and the "preferential attachment"

(PA) described briefly in Section 2.1. PA is drawn with probability α for items and β for users; otherwise, nodes are selected by UA. When PA is selected, we have to choose the fraction γ of edges that will be attached by the bouncing mechanism. More details of the bouncing mechanism will be described after the description of the CSUIM algorithm.

The procedure for generating synthetic bipartite graphs is outlined in Algorithm 1.

Algorithm 1 Cold Start User-Item Model.

```
Step 1. Initialize the graph with $m$ edges (we have $2m$ vertices).
Step 2. Add a new vertex to the graph of type user with probability $\delta$, otherwise of type item.
Step 3. Choose a neighbor to join the new vertex according to the following rules:
Step 3a. If the new node is item, then add $d_v$ edges from this node to type user vertices using the preferential attachment mechanism (with probability $\beta$) or uniform attachment (otherwise).
Step 3b. If the new node is user, then add $d_u$ edges from this node to type item vertices, using the preferential attachment mechanism (with probability $\alpha$) or uniform attachment (otherwise).
Step 3c. Consider the newly added vertex $v_0$ and edges from this node added by preferential attachment (nodes $u_i$ and $v_i$ are from different modalities). Select $\gamma$ fraction of those end nodes. For each node $u_1$ from this set, pick at random one of its neighbors, $v_2$. From the randomly selected node $v_2$, select its neighbor $u_3$ at random again. Connect the new node $v_0$ to the node $u_3$ selected in this way instead of the original node $u_1$ obtained by preferential attachment.
Step 4. Repeat Steps 2 and 3 $T$ times.
```

Step 3c emulates the behavior called recommendation. One can imagine that a customer who is going to buy one of the products encounters another consumer who already purchased it, and recommends him another product instead. The first consumer changes his/her mind and follows this recommendation with a probability of γ. By varying this parameter, one can observe what happens when people are more or less amenable to the recommendation.

Selecting products by uniform attachment simulates consumers that do not bother about which product to choose. Preferential attachment simulates consumers that look for products on their own (e.g., dresses unseen frequently on the street). Note that this model of graph growth simulates a very special kind of purchase behavior—namely, the behavior of only new consumers and new products. Despite its limited applicability, the model is very important, because it concentrates on a very hard part of the recommendation process called "cold start". Cold start concerns the issue that the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information. Recommender systems form a specific type of information filtering (IF) technique that attempts to present information items (e.g., movies, music, books, news, images, web pages) that are likely of interest to the user. Typically, a recommender system compares the user's profile to some reference characteristics. These characteristics may be from the information item (the content-based approach) or the user's social environment (the collaborative filtering approach). More detailed specifics of this hard problem and some solutions have been presented in [36,37].

It is easy to see that after t iterations with the bouncing mechanism disabled ($\gamma = 0$), we have $|U(t)| = m + \delta t$ vertices of type *user* and $|V(t)| = m + (1 - \delta)t$ vertices of type *item*. The average number of edges attached in one iteration is $\eta = d_u\delta + (1 - \delta)d_v$. After a large number of iterations, we can skip m initial edges in further calculations. Thus, we can show that the average number of vertices of type *user* and of type *item* depends only on iteration t and δ and does not depend on m, d_v, or d_u. The total number of edges depends only on d_v, d_u, and δ. This is not good news, because we cannot use them to estimate all parameters of the generator, especially β, α, and γ.

In the next section, an approach and a method of parameter extraction based not on the current state but rather on the dynamics of the network is presented.

4. Motivation for Proposed Approach

One can ask, how do you estimate generator parameters? Any approach to network parameter estimation should be based on the observable quantities that should be turned to the model parameters.

In the model described above, we can essentially observe the nodes and their interconnection, as well as their statistics (like degree distributions and/or clustering coefficients) as a source of information for parameter estimation.

At least three types of approaches seem to be considered:

- Analytical;
- Machine-learning; and
- Brute force.

An analytical approach would mean establishing a closed-form model for some of the observables, such as node degree distribution in both modalities and an attempt to solve it analytically for the parameters. As we will see in the next section, the differential equation for the node degree distribution is not simple to solve, and only approximate solutions are known in the literature for particular settings of the variables, even in the simple case of no recommendation ($\gamma = 0$).

A machine-learning approach was applied in [38], but there seems to be no simple relationship to be extracted via machine learning.

Finally, a brute force approach would be to slice the space of parameters and then to generate a sample for each of the parameter space slices, compute the observables from the sample, and to choose the parameter set for which the sample is closest to the real graph.

Eventually, we follow the last path; however, we simplify the process. In the simplified process, we exploit independences between some effects of the parameters, as well as some simplifying implications of the theoretical models.

5. Theoretical Node Degree Distribution Models

As indicated in the previous section, in our approach, we measure some characteristics of a network to estimate the parameters of the model. One of the most important properties of a network is the node degree distribution for each modality. We consider the probability that a node has degree k at some moment in time t and denote it as $p_k(t)$. Variable t can be interpreted as a number of iterations made while generating a graph from the model.

Let us concentrate on CSUIM (see Algorithm 1 from Section 3) when there is no bouncing mechanism. The bouncing mechanism is disabled when $\gamma = 0$. Let ζ_g represent the rate at which new nodes are introduced in modality g (items or users) that is, $\zeta_g \Delta t$ nodes of modality g are added in time interval of duration Δt. In the current model $\zeta_{users} = \delta, \zeta_{items} = 1 - \delta$ (on average) is added in a single time interval. Let $N_{k,g}(t)$ denote the expected number of nodes of modality g whose degree is k at time t. Let us consider multiple attachments. Each new node of modality g that is introduced chooses $\theta_{\overline{g}}$ existing nodes ($\theta_{users} = d_u$, $\theta_{items} = d_v$)of opposite modality $\overline{g}$. With $\theta_{n,g}$, let us denote the number of nodes attached to a new node of modality $\overline{g}$ using the non-preferential (uniform) attachment, and with $\theta_{p,g}$, let us denote the number of nodes attached to a new node of modality $\overline{g}$ using preferential attachment ($\theta_{p,users} = \beta d_u$, $\theta_{p,items} = \alpha d_v$, $\theta_{n,users} = (1-\beta) d_u$, $\theta_{n,items} = (1-\alpha) d_v$,).

Following the argument from [35], one can see that the node distribution over time is governed by the equation:

$$
\begin{aligned}
\dot{N_{k,g}} = & \frac{\zeta_{\overline{g}} \theta_{p,g}}{\sum_{\ell} \ell N_{\ell,g}(t)} ((k-1) N_{k-1,g}(t) - k N_{k,g}(t)) \\
& + \frac{\theta_{n,g} \zeta_{\overline{g}}}{N_g(0) + \zeta_g t} \left(N_{k-1,g}(t) - N_{k,g}(t) \right) \\
& + \zeta_g \delta_{k,\theta_{\overline{g}}}
\end{aligned}
\quad (3)
$$

with $\sum_\ell \ell N_{\ell,g}(t) = (\zeta_{\overline{g}}\theta_g + \zeta_g\theta_{\overline{g}})t$.

In [35], the solutions for the extreme cases of $\theta_g = \theta_{n,g}$ (pure uniform attachment) and $\theta_g = \theta_{p,g}$ (pure preferential attachment) were found. It turns out that for t tending to infinity, the node distributions are governed approximately by exponential distribution and power distribution resp.

The change of $N_k(t)$ for pure uniform attachment is given by:

$$\frac{N_k(t+\Delta t) - N_k(t)}{\Delta t} = \frac{\theta_{n,g}\frac{\zeta_{\overline{g}}}{\zeta_g}\zeta_g}{N(0)+\zeta_g t}(N_{k-1} - N_k) + \zeta_g \delta_{k,\theta_{n,\overline{g}}}. \tag{4}$$

An approximate solution tends to the following when time is going to infinity:

$$p_{k,UFR}(t) \approx \frac{1}{\theta_{n,g}\frac{\zeta_{\overline{g}}}{\zeta_g}} \left(\frac{\theta_{n,g}\frac{\zeta_{\overline{g}}}{\zeta_g}}{\theta_{n,g}\frac{\zeta_{\overline{g}}}{\zeta_g}+1} \right)^{k-\theta_{n,g}\frac{\zeta_{\overline{g}}}{\zeta_g}+1} u(k-\theta_{n,\overline{g}}). \tag{5}$$

In the case of preferential attachment, each newly attached node adds one to N_{β_p} at that instant. Then, $N_k(t)$ evolves according to the equation:

$$\dot{N_k} = \frac{\zeta_{\overline{g}}\theta_{n,g}\frac{\zeta_g}{\zeta_{\overline{g}}}}{\sum_\ell \ell N_\ell}((k-1)N_{k-1} - kN_k) + \zeta_{\overline{g}}\delta_{k,\theta_{p,\overline{g}}} \tag{6}$$

$\sum_\ell \ell N_\ell = (\zeta_{\overline{g}}\theta_g + \zeta_g\theta_{\overline{g}})t$

$$\lim_{t\to\infty} p_{k,PFR}(t) = \frac{\frac{(\zeta_{\overline{g}}\theta_g+\zeta_g\theta_{\overline{g}})}{\zeta_{\overline{g}}}\theta_{n,g}\frac{\zeta_g}{\zeta_{\overline{g}}}(\theta_{n,g}\frac{\zeta_g}{\zeta_{\overline{g}}}+1)}{k(k+1)(k+2)} u(k-\theta_{p,\overline{g}}). \tag{7}$$

However, no mixed case was considered in [35]. The mixed case was treated by [5], though only for large k. It turns out that the distribution in the mixed case is approximately power distribution, though with a complex exponent. The formula in [5] is derived using the relaxation of the degree to a real positive number, defining probability density function over degrees. Using our notation, we have the following equation:

$$\Phi\{k_g(t) < k\} = 1 - \left(\frac{(1-\frac{\theta_{p,g}}{\theta_g})\eta + \zeta_g\frac{\theta_{p,g}}{\theta_g}k}{(1-\frac{\theta_{p,g}}{\theta_g})\eta + \zeta_g\frac{\theta_{p,g}}{\theta_g}\theta_g} \right)^{\frac{-\eta}{(1-\zeta_g)\frac{\theta_{p,g}}{\theta_g}\theta_{\overline{g}}}}, \tag{8}$$

where $\Phi\{k_g(t) < k\}$ is the probability that modality g vertex g has degree k_g, which is less than threshold value k, and $\eta = d_u\delta + (1-\delta)d_v$ is the average number of edges attached in one iteration.

Thus, we get

$$\begin{aligned} p_{k,UIM} = & \frac{\eta}{(1-\zeta_g)\frac{\theta_{p,g}}{\theta_g}\theta_{\overline{g}}}\zeta_g\frac{\theta_{p,g}}{\theta_g} \\ & \left(\frac{(1-\frac{\theta_{p,g}}{\theta_g})\eta + \zeta_g\frac{\theta_{p,g}}{\theta_g}k}{(1-\frac{\theta_{p,g}}{\theta_g})\eta + \zeta_g\frac{\theta_{p,g}}{\theta_g}\theta_g} \right)^{\frac{-\eta}{(1-\zeta_g)\frac{\theta_{p,g}}{\theta_g}\theta_{\overline{g}}}} u(k-\theta_{\overline{g}}). \end{aligned} \tag{9}$$

In our paper [38], we tried to extract the α and β coefficients from formula (9), but it did not match the experimental distribution well.

Therefore, we seeked an alternative to this. This alternative is shown in the sections below.

6. Our Approach to Parameter Estimation

The model from the previous section, though difficult enough for an analytical solution, still means a substantial simplification in that γ is set to zero, and we deal with time tending to infinity and assume the k is large.

So, first of all, why shall we assume that $\gamma = 0$? If there is no bouncing, then we can easily see that the node degree distributions of both modalities are independent of one another so that they can be considered separately. Also, as α and β apparently influence one or the other modality degree distribution, we can guess that both can be estimated separately. This reduces the search space drastically, but what will happen if $\gamma > 0$? In this case, "under a stable distribution", two nodes of, say, user type will pick up item nodes from approximately the same distribution. So if bouncing occurs, then it is equally likely that a node of degree k will increase its degree instead of a node of degree l, and that something will happen in the reverse direction. So, we can expect that under "modest" values of γ, the marginal distributions of degrees of both modalities will remain unchanged, and a model with $\gamma = 0$ is justifiable for them.

However, we can easily guess that γ will impact the clustering measures. So that after estimating α, β, we can estimate γ separately.

7. A Linear Relationship to Obtain α and β

We would like to demonstrate how probability p_k of a node having degree k changes in CSUIM. With respect to the definition of probability from Equation (9), Figure 1a,b depict dependency between $\ln(p_k)$ and $\ln k$ for fixed values of α or β, depending on modality. It turns out that for small k (consuming most of the probability mass) and fixed α (β), the value of $\ln(p_k)$ decreases nearly linearly with $\ln k$. We can see that when we add more edges to a node (ten times more), linear characteristics of the relation between $\ln(p_k)$ and $\ln k$. This gives us an insight about setting up values of d_u and d_v.

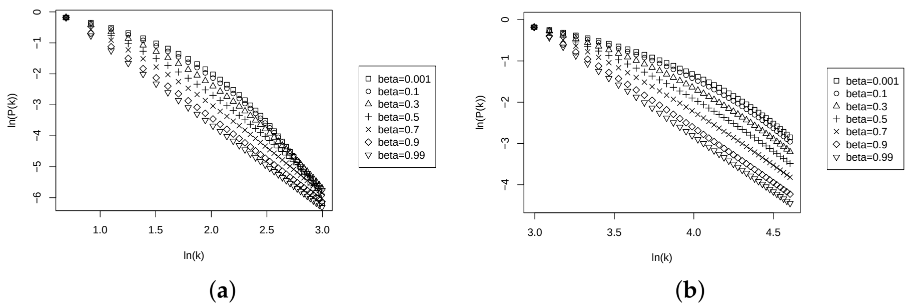


Figure 1. Plot of theoretical relation $\ln P(k)$ versus $\ln k$ for (**a**) $k = 2, 2.5, 3, ..., 20$ and $d_u = 2$, $d_v = 3$ and (**b**) $k = 20, 22, 24, ..., 100$ and $d_u = 20$, $d_v = 30$ for different values of β, where $P(k) = p_{k,UIM}$ for the user's modality. In both cases, $P(k)$ does not depend on α value.

The same dependency occurs when we consider simulations with the CSUIM (see Figure 2a,b). We can see that the linear relation between $\ln P(k)$ and $\ln k$ almost does not change when we fix β and change the value of α from 0 to 0.99. Note that Equation (11) does not contain α. This experiment shows that not only in theory, but also in practice, computing β does not depend on α value.

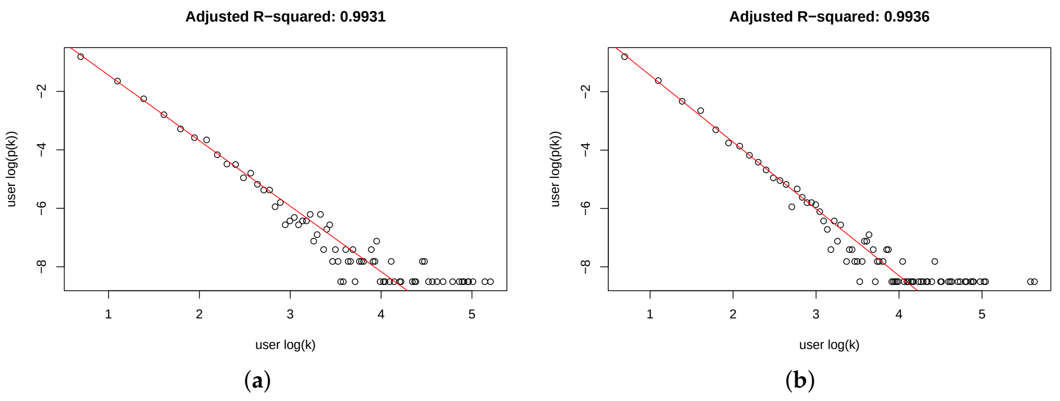


Figure 2. Plot of experimental relation $\ln P(k)$ versus $\ln k$ for generated graph for modality users with a different α value: in (**a**) $\alpha = 0$ and in (**b**) $\alpha = 0.99$. Other parameters are the same in both cases: 10K iterations, $\beta = 0.99$, $\delta = 0.5$ and $d_u = 2$, $d_v = 3$. The red line is the regression line based on this relation for $k = d_u, d_u + 1, ..., 2(d_u + d_v)$ which contains most of the distribution mass.

Therefore, we looked at the relationship between α (analogously for β) and the direction coefficient of the straight line approximating the relationship between $\ln(p_k)$ and $\ln k$, and drew it for various values of α (β). We see that for a wide range of values of α (β), this relationship is linear, both for the theoretical and simulation models.

This insight led us to the algorithms for the identification of α and β, as described below.

How can it be explained that α and β are linearly dependent on the degree distribution exponent?

As already mentioned, when α or β (for the respective modality) is set to 1, then we have to do with the preferential attachment for that modality and the degree distribution follows a power-law, whereas when set to 0, the exponential distribution is followed. For values in-between, we have to do with a kind of mixture of both (which seems not to be a simple one).

If we take the formula $\ln P(k) / \ln k$ and draw it for various values of k as a function of β, we will see that in a large range of values there is a nearly linear relationship. This result is shown in Figure 3. Therefore, we exploited it for an estimation of β.

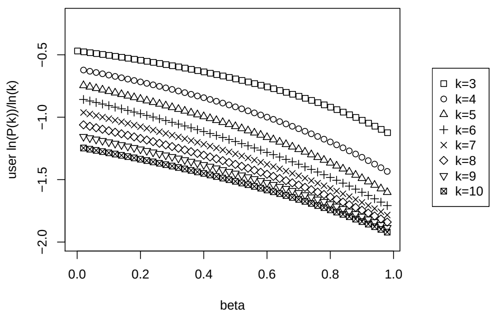


Figure 3. Plot of theoretical relation $\ln P(k) / \ln k$ for $k = 3, ..., 10$ and $d_u = 2$, $d_v = 3$, where $P(k) = p_{k,UIM}$ for the user's modality. In this case, $\ln P(k) / \ln k$ does not depend on the α value.

In the CSUI model when β grows, the probability of connecting the new link with preferential attachment grows as well. Thus, we can approximate the distribution of vertices' degrees by the power-law distribution from the experimental degree distribution and compute the exponent of this distribution. We have

$$p(k) = \exp(b) \cdot k^a. \tag{10}$$

After applying the ln function to both sites, we get:

$$\ln(p(k)) = a \cdot \ln(k) + b. \tag{11}$$

We see in Figure 4 that theoretically for different combinations of d_u and d_v, parameter β has a linear relationship with exponent (a coefficient in Equation (11)) of the power-law distribution of a vertex degree. This observation provides us with an algorithm for β parameter estimation. Analogously, we can estimate a α parameter from the exponent of distribution of vertices from the item modality. Moreover, when β grows up to 1 (preferential attachment), then we get the desired power-law distribution of nodes degree $P(k) \propto k^{-3}$.

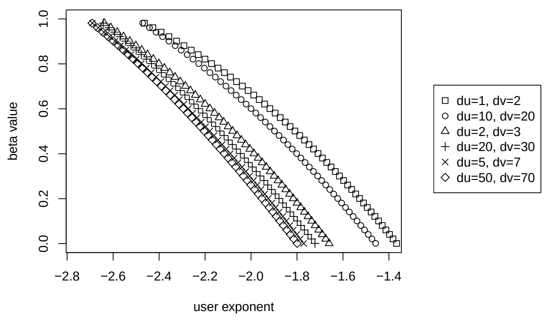


Figure 4. Plot of theoretical exponent of power-law degree distribution for different d_u and d_v for modality users. Degrees k taken for estimation are from d_u to $2(d_u + d_v)$. Those degrees have most of the distribution mass.

The preferential attachment has a power-law distribution with a "heavy tail" of node degrees, and the uniform attachment has an exponential distribution of node degrees, with a "light tail". As demonstrated in [39], an empirical mixture of these two distributions can be approximated with the power-law distribution. Therefore, linear regression analysis has been used sometimes to evaluate the fit of the power-law distribution to data and to estimate the value of the exponent. This way, one can also obtain the mixture parameter, α. The rationale behind this approach is that the heavy tail distribution dominates over the exponential distribution for nodes of higher degree. This technique produces biased estimates (see [39]). As we see in the experiments, it is unreliable for low values of α (β) (below 0.1)—see Figure 5a ($\ln P(k)$ vs $\ln(k)$) and Figure 5b ($\ln P(k)$ vs k).

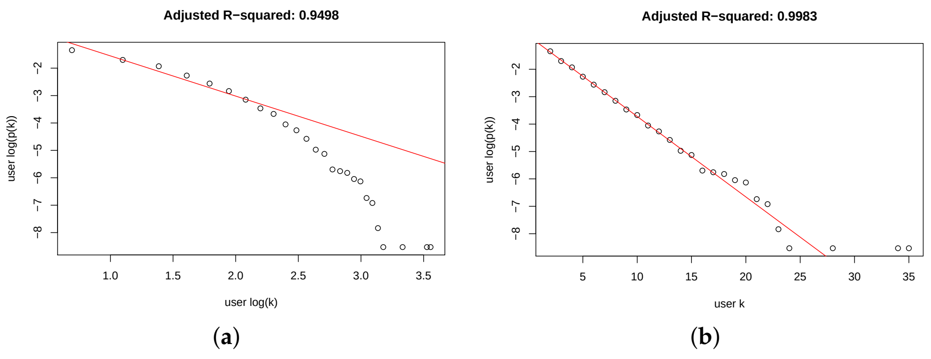


Figure 5. Plot of experimental relation $\ln P(k)$ versus $\ln k$ (**a**) and versus k (**b**) for generated graph for modality users in 10K iterations, $\alpha = 0.02$, $\beta = 0.02$, $\delta = 0.5$, and $d_u = 2$, $d_v = 3$, where $P(k) = p_{k,UIM}$. Drawn line is regression line based on this relation for $k = d_u, d_u + 1, ..., 2(d_u + d_v)$, which contains most of the distribution mass.

8. A Linear Relationship for γ

The bouncing parameter of the graph model may be used to model the behavior of users vulnerable to recommendations. We find out that this parameter is linearly correlated with a graph metric called "optimal modularity" (see [9]).

Modularity is a measure of the quality of the clustering of nodes in a graph (we describe it briefly below). *Optimal modularity* is the modularity of such a clustering of nodes for which the modularity is the highest among all the node clusterings of a given graph. It is known that finding the optimal modularity is an NP-hard task; therefore, there exist various greedy algorithms without a range guarantee. So, in fact, this term should be called "the optimal modularity for the algorithm X", and so we mean here the optimal modularity computed by the algorithm described in [9].

Modularity is the fraction of the edges that fall within the given groups (clusters) minus the expected such fraction if edges are distributed at random. The value of the modularity lies in the range $[-\frac{1}{2}, 1]$. It is positive if the number of edges within groups exceeds the number expected on the basis of chance. Examples of graph clusterings with positive and negative modularity values are shown in Figures 6a,b, respectively. The upper boundary (modularity=1) is approached if one has a multitude of complete graphs. For a given division of the network's vertices into some clusters (called groups, communities, or modules), modularity reflects the concentration of nodes within modules compared to a random distribution of links between all nodes regardless of modules.

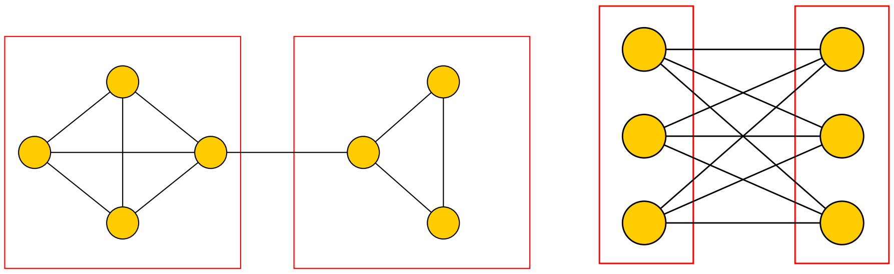

(**a**) (**b**)

Figure 6. (**a**) Simple case when modularity is positive near zero. (**b**) Simple case when modularity is negative.

There are many ways to express the modularity. In our approach, we compute Newman's modularity (see [9]) as follows:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i * k_j}{2m} \right] \delta_K(c_i, c_j), \tag{12}$$

where A_{ij} represents the adjacency matrix, $A_{ij} = 1$ when there is an edge between nodes i and j and 0 otherwise, $k_i = \sum_j A_{ij}$ is the sum of the weights of the edges attached to the vertex i, c_i is the community to which the vertex i is assigned, the δ-function is Kronecker delta, $\delta_K(u, v) = 1$ iff $u = v$ and 0 otherwise, and $m = \frac{1}{2} \sum_{ij} A_{ij}$. The above formula for modularity can also be expressed as the difference between the quotient of the number of edges inside of communities and of the total number of edges minus the sum of squares of the shares of edges that have at least one end in the community.

In our computations of the optimal modularity, communities are obtained based on the Newman's modularity concept. The algorithm runs as follows: initially, each node constitutes its own community, then nodes are moved between neighboring communities until a stopping criterion is reached. The obtained communities receive distinct identifiers called a modularity class. A node is moved to the community of one of its neighbors if this would increase the modularity of the entire network. At each step, the node giving the maximum modularity gain is selected. The process is terminated if no gain of modularity can be achieved.

Part of the Newman's algorithm efficiency (see [9]) results from the fact that the gain in modularity ΔQ obtained by moving an isolated node i into a community C can easily be computed by:

$$\begin{aligned}\Delta Q = &\left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] \\ &- \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]\end{aligned} \tag{13}$$

where $\sum_{in}$ is the sum of the weights of the links inside C, $\sum_{tot}$ is the sum of the weights of the links incident to nodes in C, k_i is the sum of the weights of the links incident to node i, $k_{i,in}$ is the sum of the weights of the links from i to nodes in C, and m is the sum of the weights of all the links in the network. A similar expression is used in order to evaluate the change of modularity when i is removed from its community. Therefore, in practice, one evaluates the change of modularity by removing i from its community and then by moving it into a neighboring community.

To sum up, the Newman's optimal modularity tells us very important thing—how much our graph differs from a random one. In a fully random graph, edges are attached to some nodes at random from some distributions. The bouncing parameter γ of the CSUI model gives us a kind of dependence of node linking to other nodes—selecting both ends of an edge. Value γ represents a fraction of edges attached in a preferential way, which were created using the bouncing mechanism. The greater the value of γ is, the stronger dependence in creating links in the graph occurs. When we have some kind of dependence while creating links, the greater the value of modularity.

Let us return to the step in the CSUI model where the new node is added, and the bouncing mechanism is active. Let us consider bouncing from the newly created user vertex u (see Figure 7). Firstly, the bouncing algorithm selects an item vertex, i. From this vertex, we can go further to user modality through edges added in previous steps either by an edge added in one of the previous iterations by adding a user node or item node. From the fact that we deal with the power-law distribution of a vertex degree, we know that most of the distribution mass have vertices with the smallest degree. Thus, it is more probable that we go through the edge added by adding a user node u_2 and from this node to an item node i_k, which is the end node of the edge e. Thus, we created a new edge, (u, i_k). So the probability of creating edge (u, i_k) is:

$$P(i_k|u) \approx \sum_{u_2,i} P(i|u)P(u_2|i)P(i_k|u_2). \tag{14}$$

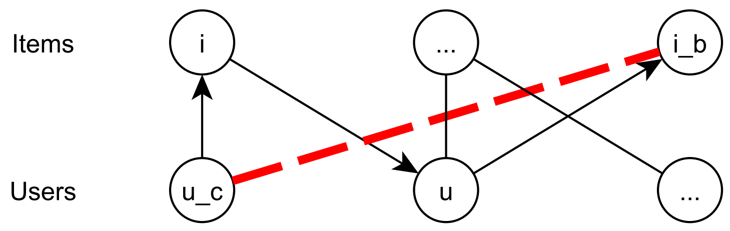


Figure 7. Example of creating a new edge (red dashed line) from new node u using a bouncing mechanism. Directed arrows indicate the following steps of a bouncing mechanism in an undirected bipartite graph.

Equation (14) can be written in this form, because if we add a new vertex u to the graph, then outgoing edges from this node are independent of each other. Because the node i has a low degree, most of the outgoing links from i_k are independent, and analogously, most of the user nodes are of low degree, so outgoing links are independent. In general, after "sufficient" time during the further evolution of the network, we get $P(i_k|u_2) = P(u_2|i_k)$, so we have:

$$\begin{aligned} P(i_k|u) &= \sum_{u_2,i} P(i|u)P(u_2|i)P(i_k|u_2) \\ &= P(i_k|u_2) \underbrace{\sum_{u_2,i} P(i|u)P(u_2|i)}_{=1} \\ &= P(i_k|u_2). \end{aligned} \tag{15}$$

Thus, the bouncing mechanism does not change distribution on most degrees (small degree) and can be considered separately from α and β parameters.

On the other hand, modularity is a measure of distribution change of edge placement in graphs compared to their random placement. Edges mentioned before are placed almost randomly, and they have no influence on the modularity value. However, there are other combinations of the placement in bouncing—some edges are added when we added edges from u_2 and also u. In this case, edges are not independent because adding an edge from u_2 to i increases the probability of adding an edge from u to i. Thus, the independence of distribution is distorted, which implies a change of modularity. Therefore, we conclude that there may be a way to identify the bouncing parameter from the modularity, and we will determine this relationship empirically. In Figure 8a, we see that this relationship seems to be linear even for small values of α and β. Unfortunately, when we add more edges in one step, the linear relation gets weaker—see Figure 8b.

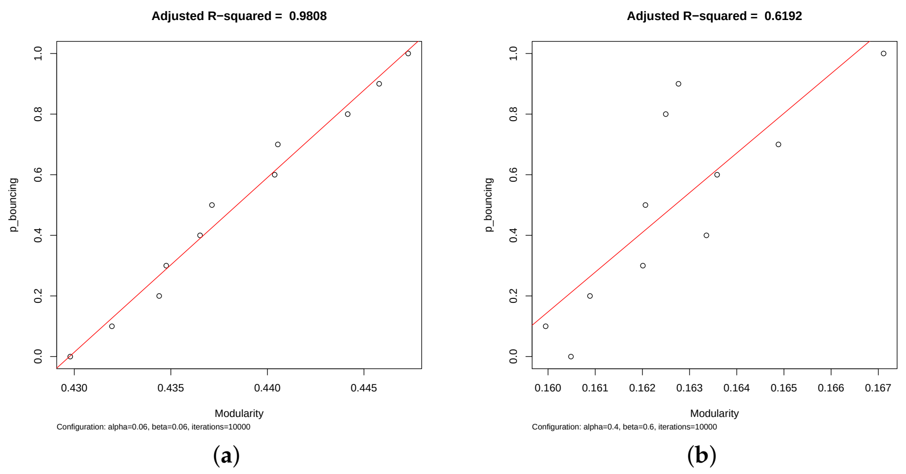


Figure 8. Plot of modularity for prediction of γ. Test setting in (**a**): 10,000 iterations, $d_u = 2$, $d_v = 3$, $\alpha = 0.06$, $\beta = 0.06$, and $\delta = 0.5$. Test setting in (**b**): 10,000 iterations, $d_u = 10$, $d_v = 20$, $\alpha = 0.4$, $\beta = 0.6$, and $\delta = 0.5$. Drawn line is regression line.

9. Parameter estimation

Here, we estimate the parameters of the model based on a couple of observable network properties. The estimations are based on theoretical relationships between model parameters and metrics from the generated network from the previous section. In this section, we propose algorithms for the computation of all CSUI models. First, we describe the retrieval of parameters δ, m, d_u, and d_v. Then, we propose two algorithms. The first algorithm estimates α and β parameters using the distribution of node degree in each modality and linear regression. The second one uses modularity measure and linear regression for computation of the γ parameter.

9.1. Parameter δ

Theoretical equations from the previous section after some modification are useful to estimate parameters of a bipartite graph generator. The simplest one is δ, which is the probability that a new vertex v added to the graph in iteration t is a user $v \in U$, so $1 - \delta$ means the probability that the new vertex v is an item $v \in V$.

$$\delta = \frac{|U|}{|U \cup V|}, \tag{16}$$

where $|U|$ cardinality is the set of nodes users and V is the set of nodes of item type.

9.2. Parameters d_u, d_v and m

There are two approaches to obtaining d_u and d_v. The first and simplest one is to set d_u as the minimal degree in the user set, and to analogously set d_v as the minimal degree in the item set.

The second way is more complicated. The average number of edges attached in one iteration is $\eta = d_u\delta + (1-\delta)d_v$. η is easy to estimate from graph as $\eta = \frac{|E|}{|U \cup V|}$, where $|E|$ is the total number of edges in the graph. δ is computed from the previous section. Most of the vertex degree distribution mass is on the lower degrees k, so we can make the integer minimization of $d_u + d_v$ with an additional restriction $d_u\delta + (1-\delta)d_v - \eta = 0$. Another way is the brute force approach. It is done based on vertex degree distribution in each modality. We fix some d_u and compute the value d_v from equation $|E| = d_u \cdot |U| + d_v \cdot |V|$.

m is the number of initial edges. It must be at least $\max(d_u, d_v)$. The better way is to set it for computation on $d_u + d_v$ because it speeds up a few initial steps when there are few nodes in a graph.

9.3. Calculations of α and β

In Section 7, we had shown theoretical linear relationships for obtaining α and β. Therefore, we can compute α and β from linear models:

$$\alpha = a_1 \cdot exp_{item} + a_0, \tag{17}$$

where a_1, a_0 are some constants calculated from the linear regression model and exp_{item} is an exponent of the power-law distribution of node degree in item modality. Analogously, for the β parameter, we obtain

$$\beta = b_1 \cdot exp_{user} + b_0, \tag{18}$$

where b_1, b_0 are some constants calculated from the linear regression model and exp_{user} is an exponent of the power-law distribution of node degree in user modality. Technical details of computing exponent of the power-law distribution of node degree are shown in Sections 9.4 and 9.5.

9.4. Calculations of $\log_{pk}$ and $\log_k$

In this section, we show how to compute an empirical power-law distribution. For each modality, we have a two-dimensional array $deg_k[max_k][2]$, where max_k is maximal degree in the considered modality. Based on this array, we compute arrays: $\log_{pk}$ which contains the probability that the vertex has degree k and $\log_k$ with the logarithm of the vertex degree k. The pseudocode is shown in Algorithm 3. It is important for the Algorithm 4 from Section 9.5 that this array contains only existing node degrees.

9.5. Calculations of Power-Law Exponent

For each modality in the graph, we compute the exponent of the power-law distribution in the following manner. We have two arrays computed in Section 9.4: $\log_{pk}$, which contains the probability that the vertex has the degree k, and $\log_k$ with the logarithm of the vertex degree k. From those arrays, we compute the power-law exponent exp of the node degree distribution in Algorithm 4.

Algorithm 2 Computation of α and β.

```
Input: Bipartite graph $G = U \cup V$, where $U \cap V = \emptyset$. We will call $U$ user set and $V$ is item set.
Step 1. Compute exponent $exp_{user}$ of degree distribution of user node set $U$, and analogously, $exp_{item}$ of item node set $V$.
Step 2. Compute $\delta$ from Equation (16) and $d_u$ and $d_v$ from subsection 9.2.
Step 3. Define the set $\mathbb{A} = \{\alpha_1, ..., \alpha_I\}$ and the set $\mathbb{B} = \{\beta_1, ..., \beta_J\}$, to be called the grid of $\alpha$ and $\beta$ later.
Step 4. For each pair $(\alpha_i, \beta_j)$, generate a bipartite graph with these parameters, setting $\delta$, $d_u$ and $d_v$ as computed in Sections 9.1 and 9.2 and setting $\gamma$ to zero. From the generated graph, compute exponent $exp_{user_{ij}}$ of the degree distribution of the user set, and analogously, $exp_{item_{ij}}$ of the item set, as shown in Section 9.5.
Step 5. For the data set $D_\alpha$ consisting of pairs $(\alpha_i, exp_{item_{ij}})$, perform linear regression creating $model_\alpha$ with the response vector $\alpha$ and one predictor variable, $exp_{item}$.
Step 6. For the data set $D_\beta$ consisting of pairs $(\beta_j, exp_{user_{ij}})$, perform linear regression creating $model_\beta$ with response vector $\beta$ and one predictor variable $exp_{user}$.
Step 7. Predict $\alpha$ value from $model_\alpha$ based on $exp_{item}$ obtained from graph $G$.
Step 8. Predict $\beta$ value from $model_\beta$ based on $exp_{user}$ obtained from graph $G$.
```

Algorithm 3 Computation of arrays $\log_{pk}$ and $\log_k$.

```
Step 1. Count vertices of degrees $1, ..., max_k$, which exists in the graph, and store them in array $deg_k[i][2]$, where $deg_k[i][1]$ has the value of $k$ and $deg_k[i][2]$ contains the number of vertices with degree $k$.
Step 2. Get the count of vertices of the considered modality as $mod_{count}$.
Step 3. For each existing degree $k$ (index $i$), compute:
Step 3.1. $degree = deg_k[i][1]$
Step 3.2. $degree_{count} = deg_k[i][2]$
Step 3.3. $\log_k[i] = \log(degree)$
Step 3.4. $\log_{pk}[i] = \log(degree_{count}/mod_{count})$
Step 4. Return arrays $\log_{pk}$ and $\log_k$.
```

9.6. Calculations of γ Parameter

As we had shown in Section 10.1, this relation is well-approximated by the linear model to some extent. If $\alpha, \beta \in [0.1, 0.9]$ and $d_u, d_v \leq 5$, then the *bouncing* parameter is predicted to be quite good from the simple linear model:

$$\gamma = pb_1 \cdot modularity + pb_0, \tag{19}$$

where pb_1, pb_0 are some constants calculated from the linear regression model. Thus, we constructed the Algorithm 5.

Algorithm 4 Computation of power-law exponent exp.

```
Step 1. Fit linear model to the data: $\log_k = l_1 \cdot \log_{pk} + l_0$.
Step 2. The returned model has two coefficients: $l_0$—intercept and $l_1$—attribute coefficient.
Step 3. Get coefficient from the attribute $\log_{pk}$ and save on variable $exp$.
Step 4. Return $exp$.
```

Algorithm 5 Computation of γ.

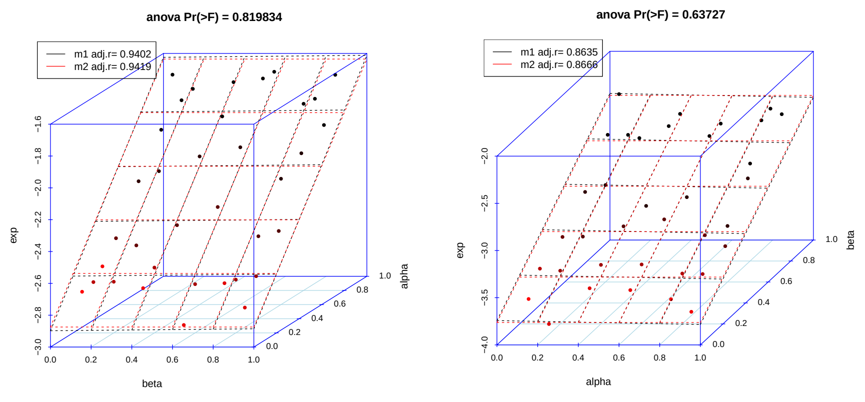
```
Input: Bipartite graph $G = U \cup V$, where $U \cap V = \emptyset$, $U$—user set and $V$—item set.
Step 1. Compute $\alpha$ and $\beta$ from Algorithm 2.
Step 2. Create grid of $\gamma_i$ values.
Step 3. For each $\gamma_i$, generate the graph model and compute modularity.
Step 4. Make dataset $D_\gamma$ containing $\gamma_i$ values and corresponding modularity values.
Step 5. Make linear regression model $model_\gamma$ having the response vector $\gamma$ and one variable $modularity$.
Step 6. Predict the $\gamma$ value from $model_\gamma$ based on $modularity$ from graph $G$.
```

10. Experimental Results

Here, we present the experimental results on parameter recovery and model quality. We performed several simulations to validate theoretical relations involving parameters α and β described in Section 7 and parameter γ described in Section 8. Those simulations are presented in Section 10.1. After verifying theoretical properties, we made parameter estimation experiments. We tested how well the parameters of the CSUI model can be obtained from several real networks. The network generated from the CSUI model and real network were compared based on a number of metrics described in Section 10.2.

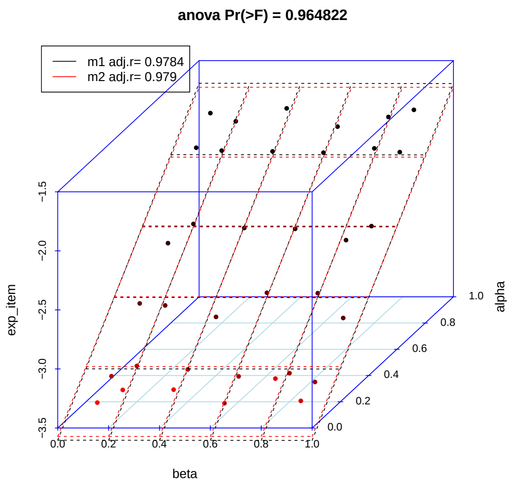
10.1. Validity of Parameter Recovery Models

In this section, we make several simulations generating networks from the CSUI model to verify theoretical properties. Experiments with α and β were made based on Algorithm 2. Experimental results in Figure 9a,b show that α and β parameters do not depend on each other. Model $m1$ contains two variables: β and exp_I in Figure 9a, α and exp_{user} in 9b. Model $m2$ contains only one variable—exp_{item} in Figure 9a, and exp_U in Figure 9b. On top of each plot is the given p-value of the ANOVA test of difference between models $m1$ and $m2$. Adjusted R-Squared values for models $m1$ and $m2$ in Figure 9a are 0.94, and the p-value of the ANOVA test is 0.82. Adjusted R-Squared values for models $m1$ and $m2$ in Figure 9b are around 0.86, and the p-value of the ANOVA test is 0.63. The p-value of the ANOVA test is greater than 0.05, so at this level of importance, there is no statistically significant difference. Thus, the α parameter does not depend on the β parameter in Figure 9a, and the β parameter does not depend on the α parameter in Figure 9b. Moreover, with more iterations (see Figure 10a,b), this independency gets stronger—thus, there is a greater value of the ANOVA test.

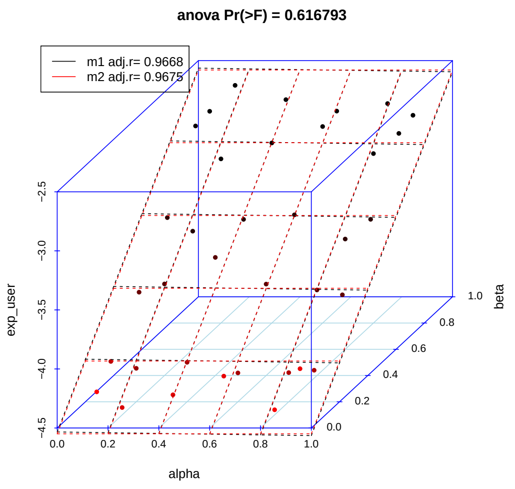


(**a**) (**b**)

Figure 9. 3D plot of the exponent of distribution of item modality exp_I (**a**) and exp_U (**b**) for prediction of parameters α (**a**) and β (**b**). Test setting: 5000 iteration, $d_u = 3$, $d_v = 2$. Adjusted R-Squared values for (**a**) are around 0.94, and for (**b**) are around 0.86. P-value of ANOVA test of difference between model $m1$ and $m2$ for (**a**) is 0.82, and for (**b**) is 0.63.



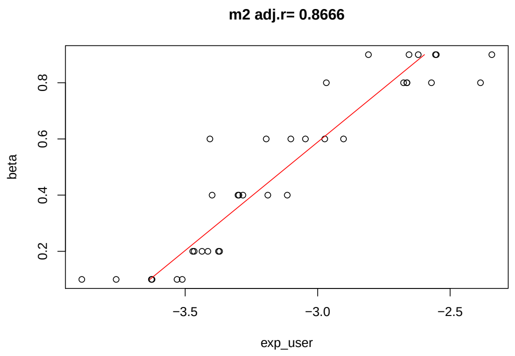
(**a**)



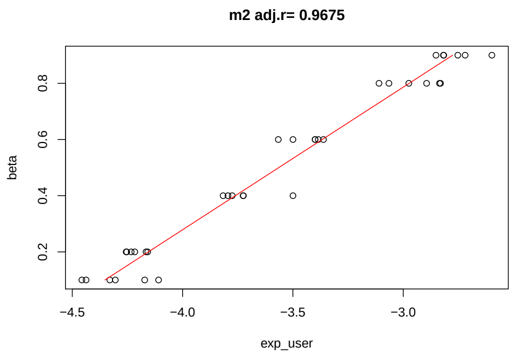
(**b**)

Figure 10. 3D plot of exponent of distribution of item modality exp_{item} (**a**) and exp_{user} (**b**) for prediction of parameter α (**a**) and β (**b**). Test setting: 50,000 iteration, $d_u = 3$, $d_v = 2$. Adjusted R-Squared values for (**a**) are around 0.98, and for (**b**) are around 0.96. P-value of ANOVA test of difference between model $m1$ and $m2$ for (**a**) is 0.96, and for (**b**) is 0.62.

The 2D plot of data obtained from the experiment is given in Figure 11a,b for 5000 and 50,000 iterations, respectively. We can see that with more iterations, the spread of points for different values of the α parameter at the same value of β is getting smaller, which gives a better prediction of the parameter β. Moreover, with more iterations, this independency gets stronger—a greater p-value of the ANOVA test. Thus, we can predict them separately.



(**a**)



(**b**)

Figure 11. 2D plot of exponent of distribution of user modality exp_{user} for prediction of parameter β. Test setting: 5000 (**a**) and 50,000 (**b**) iterations, $d_u = 3$, $d_v = 2$. Adjusted R-Squared value for (**a**) is 0.87, and in (**b**) is 0.97. Drawn line is the regression line.

Simulations with the γ parameter were made based on Algorithm 5. Plots of data obtained from the experiment for 10,000 iterations and different values of α and β are given in Figure 12. We can see an almost ideal fit (adj. R-squared value above 0.98).

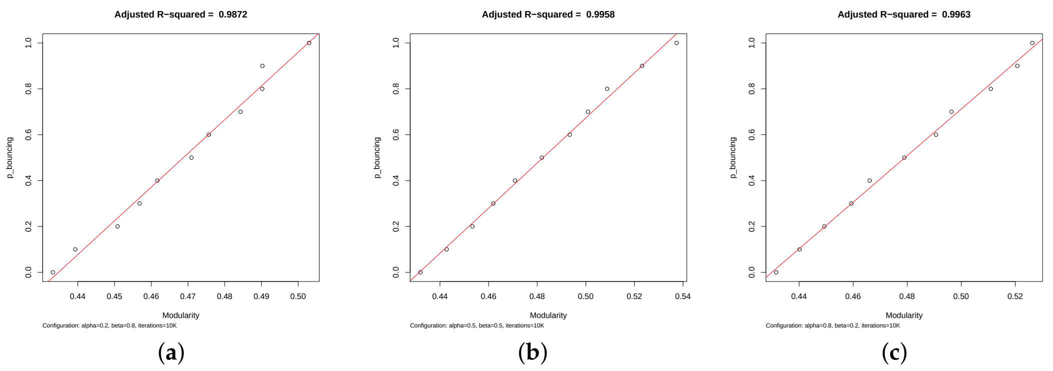


Figure 12. Plot of modularity for prediction of γ. Test setting: 10,000 iterations, $\alpha = 0.2$ and $\beta = 0.8$ (**a**), $\alpha = 0.5$ and $\beta = 0.5$ (**b**), $\alpha = 0.8$ and $\beta = 0.2$ (c). Drawn line is the regression line.

10.2. Retrieval of Parameters

In our experiments, we used several topical fora from the StackExchange data dump from December 2011. This database is available online and licensed under the Creative Commons BY-SA 3.0 license. Stack Exchange is a fast-growing network of question-and-answer sites on diverse topics, from software programming to cooking to photography and gaming. We analyzed databases from the following forums: bicycles, databaseadministrators, drupalanswers, itsecurity, physics, texlatex, theoreticalcomputerscience, unixlinux, webapplications, webmasters, and wordpress. From this data, a bipartite graph for each dataset was created. In one modality, there were users in other topics. An edge was created when a user participated in some topics by writing a post in the topic. The edge between the user and topic was created only once. We interpreted the network structure as an undirected graph with no weights per edge.

Due to the limitation of the CSUI model, we took under consideration only the giant component (GC). The giant component is the biggest connected component in a graph. In a real-world graph, GC contains 70% or more of the whole graph and influences the growth of the network. From the created bipartite graphs, we calculated several graph and model properties and compared them to an artificial graph generated from the CSUI model. Metrics used in experiments:

1. Total Nodes—the total number of nodes in GC
2. Total Edges—the total number of edges in GC
3. Average Degree—the average of node degree in GC
4. Diameter—the maximal distance between all pairs of nodes in GC.
5. Radius—The radius of GC. The radius r of a graph is the minimum eccentricity of any vertex, $r = \min_{v \in W} \epsilon(v)$. The eccentricity $\epsilon(v)$ of a vertex v is the greatest geodesic distance between v and any other vertex.
6. Average Path Length—the average number of steps along the shortest paths for all possible pairs of network nodes. It is a measure of the efficiency of information or mass transport on a network.
7. Number Of Shortest Paths—the number of shortest paths in GC
8. Communities Number—the number of communities from Neumann's modularity algorithm in GC. More details in section 8
9. Density—measures how close the network is to a complete graph. A complete graph has all possible edges and density equal to 1.
10. Modularity—the Neumann's modularity described in section 8
11. Avg Item Clustering—the average value of BLCC for modality items based on Equation (2)
12. Avg User Clustering— the average value of BLCC for modality users based on Equation (2)
13. UsersCount—the number of nodes in modality users
14. ItemsCount—the number of nodes in modality items

15. User Average Degree—the average value of users node degree
16. Item Average Degree—the average value of items node degree
17. gen alpha—the value of parameter α from the CSUI model. Computation is based on Algorithm 2 from Section 9.3. In column "Graph", it is computed based on a real graph, and in column "Model", it is computed based on a generated network from the CSUI model. This value is in $[0, 1]$ interval. We give the exact value from a linear model for demonstration purposes.
18. gen beta—the value of parameter β from the CSUI model. Computation is based on Algorithm 2 from Section 9.3. Interpretation as for the gen alpha metric.
19. gen p add user—the value of parameter δ from CSUI model. Computation is based on Section 9.1.
20. gen p bouncing—the value of parameter γ from the CSUI model. Computation is based on Algorithm 5 from Section 9.6.
21. ExpUserCoeff—the exponent of exponential distribution of node degree of modality users. Computation based on Section 9.5.
22. ExpItemCoeff—the exponent of exponential distribution of node degree of modality items. Computation based on Section 9.5.
23. graph eta—the average number of edges in one iteration, $\eta = \frac{|E|}{|U \cup V|}$.

We extracted graph parameters as shown in Section 9. It turned out (see Tables 1–5) that the most crucial parameters were d_u and d_v. Values of these two parameters determine how the graph generated by the model will be similar to a real one. We used two methods for finding optimal values d_u and d_v: discrete optimization and the brute force approach described in Section 9.2. The brute force approach gave us the best results in half of the cases.

Table 1. Experimental results for dataset *databaseadministrators* (**a**) for $d_u = 1$ and $d_v = 2$, and (**b**) for $d_u = 3$ and $d_v = 1$. "Rel. err." column is relative error.

(**a**)

Metric	Graph	Model	Rel. err.
Total Nodes	4964	4964	0.0000
Total Edges	7607	8907	0.1709
Average Degree	3.0649	3.5886	0.1709
Diameter	14	11	0.2143
Radius	8	6	0.2500
Average Path Length	4.9507	5.1501	0.0403
Number Of Shortest Paths	24636332	24636332	0.0000
Communities Number	27	34	0.2593
Density	0.0006	0.0007	0.1709
Modularity	0.6723	0.6558	0.0245
Avg Item Clustering	0.0103	0.0321	2.1292
Avg User Clustering	0.0879	0.2007	1.2837
UsersCount	1029	1015	0.0136
ItemsCount	3935	3949	0.0036
User Average Degree	7.3926	8.7754	0.1870
Item Average Degree	1.9332	2.2555	0.1667
gen alpha	1.4436	0.8392	0.4187
gen beta	0.6360	0.6413	0.0084
gen p add user	0.2073	0.2045	0.0136
gen p bouncing	1.1551	0.9586	0.1701
ExpUserCoeff	-0.9388	-0.9122	-0.0283
ExpItemCoeff	-3.4239	-4.7365	-0.3833
graph eta	1.5324	1.7943	0.1709

(**b**)

Metric	Graph	Model	Rel. err.
Total Nodes	4964	4964	0.0000
Total Edges	7607	6996	0.0803
Average Degree	3.0649	2.8187	0.0803
Diameter	14	12	0.1429
Radius	8	7	0.1250
Average Path Length	4.9507	7.0306	0.4201
Number Of Shortest Paths	24636332	24636332	0.0000
Communities Number	27	46	0.7037
Density	0.0006	0.0006	0.0803
Modularity	0.6723	0.7066	0.0511
Avg Item Clustering	0.0103	0.0006	0.9432
Avg User Clustering	0.0879	0.0049	0.9446
UsersCount	1029	1022	0.0068
ItemsCount	3935	3942	0.0018
User Average Degree	7.3926	6.8454	0.0740
Item Average Degree	1.9332	1.7747	0.0820
gen alpha	-0.4852	0.0355	-1.0731
gen beta	-0.0766	0.2028	-3.6487
gen p add user	0.2073	0.2059	0.0068
gen p bouncing	0.5000	0.2440	0.5121
ExpUserCoeff	-0.9388	-1.2235	-0.3033
ExpItemCoeff	-3.4239	-2.8832	-0.1579
graph eta	1.5324	1.4093	0.0803

Table 2. (**a**) Experimental results for dataset *bicycles* for $d_u = 2$ and $d_v = 2$. (**b**) Experimental results for dataset *drupalanswers* for $d_u = 3$ and $d_v = 1$. "Rel. err." column is relative error value.

(**a**)

Metric	Graph	Model	Rel. err.
Total Nodes	4111	4111	0.0000
Total Edges	7667	8210	0.0708
Average Degree	3.7300	3.9942	0.0708
Diameter	11	9	0.1818
Radius	6	6	0.0000
Average Path Length	4.2897	4.8359	0.1273
Number Of Shortest Paths	16896210	16896210	0.0000
Communities Number	27	31	0.1481
Density	0.0009	0.0010	0.0708
Modularity	0.5461	0.5484	0.0042
Avg Item Clustering	0.0219	0.0118	0.4614
Avg User Clustering	0.1172	0.1363	0.1626
UsersCount	636	669	0.0519
ItemsCount	3475	3442	0.0095
User Average Degree	12.0550	12.2720	0.0180
Item Average Degree	2.2063	2.3852	0.0811
gen alpha	1.2305	0.6683	0.4569
gen beta	0.5131	0.5999	0.1691
gen p add user	0.1547	0.1627	0.0519
gen p bouncing	0.3027	0.3326	0.0988
ExpUserCoeff	-0.8341	-1.0050	-0.2049
ExpItemCoeff	-3.1033	-4.3912	-0.4150
graph eta	1.8650	1.9971	0.0708

(**b**)

Metric	Graph	Model	Rel. err.
Total Nodes	6950	6950	0.0000
Total Edges	9862	9088	0.0785
Average Degree	2.8380	2.6153	0.0785
Diameter	15	12	0.2000
Radius	8	7	0.1250
Average Path Length	5.4024	7.0790	0.3103
Number Of Shortest Paths	48295550	48295550	0.0000
Communities Number	46	57	0.2391
Density	0.0004	0.0004	0.0785
Modularity	0.7090	0.7586	0.0699
Avg Item Clustering	0.0037	0.0002	0.9335
Avg User Clustering	0.0690	0.0048	0.9307
UsersCount	1071	1075	0.0037
ItemsCount	5879	5875	0.0007
User Average Degree	9.2082	8.4540	0.0819
Item Average Degree	1.6775	1.5469	0.0779
gen alpha	-0.7685	0.0916	-1.1192
gen beta	0.3099	0.3867	0.2480
gen p add user	0.1541	0.1547	0.0037
gen p bouncing	8.1270	0.0454	0.9944
ExpUserCoeff	-1.1052	-1.1859	-0.0730
ExpItemCoeff	-4.0796	-3.5429	-0.1316
graph eta	1.4190	1.3076	0.0785

Table 3. (**a**) Experimental results for dataset *itsecurity* for $d_u = 1$ and $d_v = 2$. (**b**) Experimental results for dataset *webmasters* for $d_u = 2$ and $d_v = 1$. "Rel. err." column is relative error value.

(**a**)

Metric	Graph	Model	Rel. err.
Total Nodes	5619	5619	0.0000
Total Edges	9572	10083	0.0534
Average Degree	3.4070	3.5889	0.0534
Diameter	14	11	0.2143
Radius	7	6	0.1429
Average Path Length	4.5853	4.9775	0.0855
Number Of Shortest Paths	31567542	31567542	0.0000
Communities Number	33	44	0.3333
Density	0.0006	0.0006	0.0534
Modularity	0.5976	0.5944	0.0052
Avg Item Clustering	0.0144	0.0120	0.1631
Avg User Clustering	0.0828	0.0953	0.1510
UsersCount	1148	1149	0.0009
ItemsCount	4471	4470	0.0002
User Average Degree	8.3380	8.7755	0.0525
Item Average Degree	2.1409	2.2557	0.0536
gen alpha	1.7195	0.6584	0.6171
gen beta	0.6432	0.5696	0.1144
gen p add user	0.2043	0.2045	0.0009
gen p bouncing	0.3335	0.2915	0.1261
ExpUserCoeff	-0.9237	-0.8275	-0.1042
ExpItemCoeff	-3.3054	-5.0719	-0.5344
graph eta	1.7035	1.7944	0.0534

(**b**)

Metric	Graph	Model	Rel. err.
Total Nodes	9544	9544	0.0000
Total Edges	13240	11752	0.1124
Average Degree	2.7745	2.4627	0.1124
Diameter	18	16	0.1111
Radius	9	9	0.0000
Average Path Length	5.5201	8.6383	0.5649
Number Of Shortest Paths	91078392	91078392	0.0000
Communities Number	50	69	0.3800
Density	0.0003	0.0003	0.1124
Modularity	0.7274	0.8033	0.1044
Avg Item Clustering	0.0043	0.0002	0.9598
Avg User Clustering	0.0447	0.0017	0.9630
UsersCount	2167	2214	0.0217
ItemsCount	7377	7330	0.0064
User Average Degree	6.1098	5.3080	0.1312
Item Average Degree	1.7948	1.6033	0.1067
gen alpha	-1.3478	0.5000	-1.3710
gen beta	0.2148	0.2158	0.0049
gen p add user	0.2271	0.2320	0.0217
gen p bouncing	-19.9664	-0.0550	-0.9972
ExpUserCoeff	-1.0828	-1.0952	-0.0115
ExpItemCoeff	-3.7616	-2.6420	-0.2976
graph eta	1.3873	1.2313	0.1124

Table 4. (**a**) Experimental results for dataset *theoreticalcomputerscience* for $d_u = 3$ and $d_v = 2$. (**b**) Experimental results for dataset *webapplications* for $d_u = 2$ and $d_v = 1$. "Rel. err." column is relative error value.

(**a**)

Metric	Graph	Model	Rel. err.
Total Nodes	6114	6114	0.0000
Total Edges	12744	13273	0.0415
Average Degree	4.1688	4.3418	0.0415
Diameter	12	9	0.2500
Radius	7	6	0.1429
Average Path Length	4.2949	5.1327	0.1951
Number Of Shortest Paths	37374882	37374882	0.0000
Communities Number	29	39	0.3448
Density	0.0007	0.0007	0.0415
Modularity	0.5063	0.5081	0.0036
Avg Item Clustering	0.0296	0.0098	0.6682
Avg User Clustering	0.1130	0.0873	0.2275
UsersCount	1099	1065	0.0309
ItemsCount	5015	5049	0.0068
User Average Degree	11.5960	12.4629	0.0748
Item Average Degree	2.5412	2.6288	0.0345
gen alpha	1.4205	0.7093	0.5007
gen beta	0.3262	0.2598	0.2036
gen p add user	0.1798	0.1742	0.0309
gen p bouncing	0.2097	0.2987	0.4243
ExpUserCoeff	-0.8667	-0.7816	-0.0981
ExpItemCoeff	-3.1786	-3.7817	-0.1897
graph eta	2.0844	2.1709	0.0415

(**b**)

Metric	Graph	Model	Rel. err.
Total Nodes	6831	6831	0.0000
Total Edges	8897	8525	0.0418
Average Degree	2.6049	2.4960	0.0418
Diameter	20	14	0.3000
Radius	10	8	0.2000
Average Path Length	6.1617	7.7572	0.2589
Number Of Shortest Paths	46655730	46655730	0.0000
Communities Number	52	60	0.1538
Density	0.0004	0.0004	0.0418
Modularity	0.7654	0.7915	0.0341
Avg Item Clustering	0.0025	0.0003	0.8850
Avg User Clustering	0.0250	0.0020	0.9181
UsersCount	1691	1700	0.0053
ItemsCount	5140	5131	0.0018
User Average Degree	5.2614	5.0147	0.0469
Item Average Degree	1.7309	1.6615	0.0401
gen alpha	0.5000	0.5000	0.0000
gen beta	0.2695	0.2572	0.0456
gen p add user	0.2475	0.2489	0.0053
gen p bouncing	-0.3171	-0.0321	-0.8987
ExpUserCoeff	-1.2276	-1.2472	-0.0159
ExpItemCoeff	-3.7918	-2.4565	-0.3521
graph eta	1.3024	1.2480	0.0418

Table 5. (**a**) Experimental results for dataset *texlatex* for $d_u = 1$ and $d_v = 2$. (**b**) Experimental results for dataset *wordpress* for $d_u = 3$ and $d_v = 1$. "Rel. err." column is relative error value.

(**a**)

Metric	Graph	Model	Rel. err.
Total Nodes	23668	23668	0.0000
Total Edges	44610	44443	0.0037
Average Degree	3.7696	3.7555	0.0037
Diameter	12	11	0.0833
Radius	7	6	0.1429
Average Path Length	4.4984	4.5038	0.0012
Number Of Shortest Paths	560150556	560150556	0.0000
Communities Number	38	65	0.7105
Density	0.0002	0.0002	0.0037
Modularity	0.5553	0.5587	0.0061
Avg Item Clustering	0.0102	0.0061	0.4059
Avg User Clustering	0.0997	0.1058	0.0613
UsersCount	2885	2887	0.0007
ItemsCount	20783	20781	0.0001
User Average Degree	15.4627	15.3942	0.0044
Item Average Degree	2.1465	2.1386	0.0036
gen alpha	2.0704	0.7445	0.6404
gen beta	0.8029	0.7750	0.0347
gen p add user	0.1219	0.1220	0.0007
gen p bouncing	0.2329	0.2363	0.0144
ExpUserCoeff	-0.8174	-0.8038	-0.0167
ExpItemCoeff	-4.1118	-6.2667	-0.5241
graph eta	1.8848	1.8778	0.0037

(**b**)

Metric	Graph	Model	Rel. err.
Total Nodes	17121	17121	0.0000
Total Edges	26123	22243	0.1485
Average Degree	3.0516	2.5983	0.1485
Diameter	18	14	0.2222
Radius	9	8	0.1111
Average Path Length	5.0917	7.7950	0.5309
Number Of Shortest Paths	293111520	293111520	0.0000
Communities Number	46	83	0.8043
Density	0.0002	0.0002	0.1485
Modularity	0.6683	0.7689	0.1506
Avg Item Clustering	0.0044	0.0001	0.9770
Avg User Clustering	0.0734	0.0020	0.9726
UsersCount	2557	2567	0.0039
ItemsCount	14564	14554	0.0007
User Average Degree	10.2163	8.6650	0.1518
Item Average Degree	1.7937	1.5283	0.1479
gen alpha	-0.9086	-0.3311	-0.6355
gen beta	0.2144	0.1529	0.2866
gen p add user	0.1493	0.1499	0.0039
gen p bouncing	0.5000	0.5000	0.0000
ExpUserCoeff	-0.9810	-0.9093	-0.0730
ExpItemCoeff	-4.2287	-3.6055	-0.1474
graph eta	1.5258	1.2992	0.1485

11. Conclusions

The Cold Start User-Item Model (CSUIM) of bipartite graphs is very flexible and can be applied to many different problems. In this article, we showed that the parameters of the CSUI model could be obtained easily from an unknown bipartite graph. We presented several algorithms to estimate the most important parameters:

1. δ—probability that the new vertex v added to the graph in iteration t is a user $v \in U$;
2. α—probability of *item* preferential attachment, $1 - \alpha$—probability of *item* uniform attachment;
3. β—probability of *user* preferential attachment, $1 - \beta$—probability of *user* uniform attachment;
4. γ—fraction of edges attached in a preferential way which were created using the bouncing mechanism.

We gave some advice about setting up the renaming parameters: m, d_u, and d_v. The experimental results showed that the CSUI model could be applied to some extent for modeling the bipartite graph of users and user posts.

Moreover, we gave a theoretical basis for estimating parameters α and β based on the degree distribution in each of the modalities. We showed that for small k (consuming most of the probability mass) and fixed α (or β), the value of $\ln(p_k)$ decreases nearly linearly with $\ln k$. The experiment presented in this paper proved that computing α does not depend on the β value and vice versa not only in theory, but also in practice. The sampling for linear regression models can simply be parallelized for more efficient computations. We also found out that the bouncing parameter γ was linearly correlated with Newman's optimal modularity. Experiments made on real-world graphs showed that from these theoretical relationships, the CSIU model parameters α, β, and γ could be extracted quite well.

An in-depth analysis of the CSUI model provides an essential guide to future research concerning creating disconnected graphs. In general, it is a hard problem, and to simplify it, we moded the giant component of the analyzed graph. Although the CSUI model can produce disconnected graphs by first initializing step, it can only merge disconnected components and does not produce (divide) new components, as it happens in real-world networks.

Funding: This research was funded by Ministerstwo Nauki i Szkolnictwa Wyższego as a research fellowship within Project 'Information technologies: research and their interdisciplinary applications', agreement number UDA-POKL.04.01.01-00-051/10-00.

Acknowledgments: The author would like to acknowledge all the support given by Institute of Computer Science Polish Academy of Sciences (IPI PAN) in Poland.

Conflicts of Interest: The author declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CSUIM	Cold Start User-Item Model
PDF	Probability density function
PA	Preferential attachment
UA	Uniform attachment
m	The initial number of edges, the initial number of vertices is $2m$
δ	The probability that a new vertex v added to a graph in the iteration t is a user $v \in U$, so $1-\delta$ means probability that v is an item, $v \in I$
d_u	The number of edges added from the vertex of user type in one iteration (number of items bought by a single new user)
d_v	The number of edges added from the vertex of item type in one iteration (number of users who bought the same new item)
α	The probability of *item* preferential attachment, $1-\alpha$ — the probability of *item* uniform attachment
β	The probability of *user* preferential attachment, $1-\beta$ — the probability of *user* uniform attachment
γ	The fraction of edges attached in a preferential way which were created using the bouncing mechanism
η	$\eta = d_u\delta + (1-\delta)d_v$ is the average number of edges attached in one iteration

References

1. Sharma, A. Social Networks. 2009. Available online: https://www.slideshare.net/9789189793/sharma-social-networks-68063079 (accessed on 10 February 2020).
2. Birmelé, E. A scale-free graph model based on bipartite graphs. *Discret. Appl. Math.* **2009**, *157*, 2267–2284.
3. Zheleva, E.; Sharara, H.; Getoor, L. Co-evolution of social and affiliation networks. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June 28–1 July 2009; ACM: New York, NY, USA, 2009; pp. 1007–1016, doi:10.1145/1557019.1557128.

4. Guillaume, J.L.; Latapy, M. Bipartite structure of all complex networks. *Inf. Process. Lett.* **2004**, *90*, 215–221, doi:10.1016/j.ipl.2004.03.007.
5. Chojnacki, S. Analysis of Technical Properties of Recommender Systems with Random Graphs. Ph.D. Thesis, Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland, 2012.
6. Barabasi, A. Linked - How Everything is Connected to Everything Else and What it Means for Business, Science, and Everyday Life. *Plume Books* **2003**, ISBN 978-0-452-28439-5
7. Albert, R.; Barabási, A.L. Statistical mechanics of complex networks. *Rev. Mod. Phys.* **2002**, *74*, 47–97, doi:10.1103/RevModPhys.74.47.
8. Lin, Y.R.; Chi, Y.; Zhu, S.; Sundaram, H.; Tseng, B.L. Facetnet: A framework for analyzing communities and their evolutions in dynamic networks. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008; ACM: New York, NY, USA, 2008; pp. 685–694, doi:10.1145/1367497.1367590.
9. Newman, M.E.J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2004**, *69*, 026113, doi:10.1103/PhysRevE.69.026113.
10. Leskovec, J.; Lang, K.J.; Dasgupta, A.; Mahoney, M.W. Statistical properties of community structure in large social and information networks. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008; ACM: New York, NY, USA, 2008; pp. 695–704, doi:10.1145/1367497.1367591.
11. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graphs over time: Densification laws, shrinking diameters and possible explanations. In Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, IL, USA, 21–24 August 2005; ACM: New York, NY, USA, 2005; pp. 177–187, doi:10.1145/1081870.1081893.
12. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data* **2007**, *1*, doi:10.1145/1217299.1217301.
13. Symeonidis, P.; Tiakas, E.; Manolopoulos, Y. Product Recommendation and Rating Prediction Based on Multi-modal Social Networks. In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; ACM: New York, NY, USA, 2011; pp. 61–68, doi:10.1145/2043932.2043947.
14. Mali, F.; Kronegger, L.; Doreian, P.; Ferligoj, A. Dynamic Scientific Co-Authorship Networks. In *Models of Science Dynamics*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 195–232.
15. McCallum, A. *Predictive Social Network Analysis with Multi-Modal Data*; Computer Science Department of University of Massachusetts Amherst: Amherst, MA, USA, 2004; Available online: http://helper.ipam.ucla.edu/publications/sews3/sews3_7456.pdf (accessed on 10 February 2020).
16. Kunegis, J. On the Spectral Evolution of Large Networks. Ph.D. Thesis, Institute for Web Science and Technologies, University of Koblenz-Landau, Mainz, Germany, 2011.
17. Lavia, E.F.; Chernomoretz, A.; Buldú, J.M.; Zanin, M.; Balenzuela, P. Modeling the evolution of item rating networks using time-domain preferential attachment. *Int. J. Bifurc. Chaos* **2012**, *22*, 1250180, doi:10.1142/S0218127412501805.
18. Watts, D.; Strogatz, S. Collective dynamics of 'small-world' networks. *Nature* **1998**, *393*, 440–442.
19. Erdös, P.; Rényi, A. *On the Evolution of Random Graphs*; The Mathematical Institute of the Hungarian Academy of Sciences: Hungary, Budapest, 1960; pp. 17–61.
20. Liu, Z.; Lai, Y.C.; Ye, N.; Dasgupta, P. Connectivity distribution and attack tolerance of general networks with both preferential and random attachments. *Phys. Lett. A* **2002**, *303*, 337–344, doi:10.1016/S0375-9601(02)01317-8.
21. Vázquez, A. Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations. *Phys. Rev. E* **2003**, *67*, 056104, doi:10.1103/PhysRevE.67.056104.
22. White, D.R.; Kejzar, N.; Tsallis, C.; Farmer, D.; White, S. Generative model for feedback networks. *Phys. Rev. E* **2006**, *73*, 161119, doi:10.1103/PhysRevE.73.016119.
23. Vázquez, A. Disordered networks generated by recursive searches. *EPL (Europhys. Lett.)* **2001**, *54*, 430.
24. Ahmedi, L.; Rrmoku, K.; Sylejmani, K.; Shabani, D. A bimodal social network analysis to recommend points of interest to tourists. *Soc. Netw. Anal. Min.* **2017**, *7*, 14, doi:10.1007/s13278-017-0431-8.
25. Krebs, V.E. Uncloaking Terrorist Networks. *First Monday* **2002**, *7*, doi:10.5210/fm.v7i4.941.

26. Lattanzi, S.; Sivakumar, D. Affiliation Networks. In Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; ACM: New York, NY, USA, 2009; pp. 427–434, doi:10.1145/1536414.1536474.
27. He, X.; Gao, M.; Kan, M.; Wang, D. BiRank: Towards Ranking on Bipartite Graphs. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 57–71, doi:10.1109/TKDE.2016.2611584.
28. Shi, S.; Zhang, M.; Liu, Y.; Ma, S. Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, 22–26 October 2018; pp. 127–136, doi:10.1145/3269206.3271710.
29. Cheng, Z.; Chang, X.; Zhu, L.; Catherine Kanjirathinkal, R.; Kankanhalli, M.S. MMALFM: Explainable Recommendation by Leveraging Reviews and Images. *ACM Trans. Inf. Syst.* **2019**, *37*, 16:1–16:28, doi:10.1145/3291060.
30. Ozsoy, M.G. From Word Embeddings to Item Recommendation. *arXiv* **2016**, arXiv:abs/1601.01356.
31. Vasile, F.; Smirnova, E.; Conneau, A. Meta-Prod2Vec:Product Embeddings Using Side-Information for Recommendation. *arXiv* **2016**, arXiv:abs/1607.07326.
32. Kang, Y.; Yu, N. Soft-Constraint Based Online LDA for Community Recommendation. In Proceedings of the 11th Pacific Rim Conference on Multimedia, Shanghai, China, 21–24 September 2010; pp. 494–505, doi:10.1007/978-3-642-15696-0_46.
33. Liu, C.; Jin, T.; Hoi, S.; Zhao, P.; Sun, J. Collaborative topic regression for online recommender systems: An online and Bayesian approach. *Mach. Learn.* **2017**, *106*, 651–670, doi:10.1007/s10994-016-5599-z.
34. Stan, J.; Muhlenbach, F.; Largeron, C. Recommender Systems using Social Network Analysis: Challenges and Future Trends. In *Encyclopedia of Social Network Analysis and Mining*; Alhajj, R., Rokne, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 1–22.
35. Fotouhi, B.; Rabbat, M.G. Network Growth with Arbitrary Initial Conditions: Analytical Results for Uniform and Preferential Attachment. *arXiv* **2012**, arXiv:abs/1212.0435.
36. Schein, A.I.; Popescul, A.; Ungar, L.H.; Pennock, D.M. Methods and Metrics for Cold-start Recommendations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, 11–15 August 2002; ACM: New York, NY, USA, 2002; pp. 253–260, doi:10.1145/564376.564421.
37. Lam, X.N.; Vu, T.; Le, T.D.; Duong, A.D. Addressing Cold-start Problem in Recommendation Systems. In Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, Suwon, Korea, 15–16 January 2008; ACM: New York, NY, USA, 2008; pp. 208–211, doi:10.1145/1352793.1352837.
38. Kłopotek, R.A. Study on the Estimation of the Bipartite Graph Generator Parameters. *Language Processing and Intelligent Information Systems*; Kłopotek, M.A., Koronacki, J., Marciniak, M., Mykowiecka, A., Wierzchoń, S.T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7912, pp. 234–244, doi:10.1007/978-3-642-38634-3_26.
39. Shatnawi, R.; Althebyan, Q. An Empirical Study of the Effect of power-law Distribution on the Interpretation of OO Metrics. *ISRN Softw. Eng.* **2013**, *2013*, 198937, doi:10.1155/2013/198937.

© 2020 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).