Enhancing Long Tail Item Recommendations Using Tripartite Graphs and Markov Process

Joseph Johnson
3361 TMCB
Computer Science Department
Brigham Young University
Provo, Utah 84602, USA
josephjohnson11@gmail.com

Yiu-Kai Ng
3361 TMCB
Computer Science Department
Brigham Young University
Provo, Utah 84602, USA
ng@compsci.byu.edu

ABSTRACT

Given that the Internet and sophisticated transportation networks have made an increasingly huge number of products and services available to the public, consumers are unable to identify, much less evaluate the usefulness of, such goods accessible to them. Modern recommendation systems filter out products of lesser utility to the customer, showcasing those items of higher preference to the user. While current state-of-the-art recommendation systems perform fairly well, they generally do better at recommending the popular subset of all products available rather than matching consumers with the vast amount of niche products in what has been termed the "Long Tail". In their seminal work, "Challenging the Long Tail Recommendation", Yin et al. make an eloquent argument that the long tail is where organizations can create the most value for their consumers. They also argue that existing recommender systems operate fundamentally different for long tail products than for mainstream goods. While matrix factorization, nearest-neighbors, and clustering work well for the "head" market, the long tail is better represented by a graph, specifically a bipartite graph that connects a set of users to a set of goods. In this paper, we discuss the algorithms presented by Yin et al., as well as a set of similar algorithms proposed by Shang et al., which traverse the bipartite graphs through a random walker in order to identify similar users and products. We build on elements from each work, as well as elements from a Markov process, to facilitate the random walker's traversal of tripartitle graphs into the long tail regions. This method specifically constructs paths into regions of the long tail that are favorable to users.

KEYWORDS

Long tail recommendation, tripartite graphs, Markov process

ACM Reference format:
Joseph Johnson and Yiu-Kai Ng. 2017. Enhancing Long Tail Item Recommendations Using Tripartite Graphs and Markov Process. In *Proceedings of WI'17, Germany, Aug 17,* 8 pages.
DOI: 10.475/123_4

WI'17, Germany

DOI: 10.475/123_4

1 INTRODUCTION

Majority of the value that recommendation systems provide comes through effective recommendations of the long-tail. A superior recommendation system can be reduced to a superior algorithm for long tail recommendations for two reasons: 1) long tail products lead to higher profitability [19], and 2) long tail recommendations are a more difficult problem to solve [14].

1.1 Profitability of Long Tail Recommendations

In his groundbreaking book, "The Long Tail: Why the Future of Business is Selling Less of More" [2], Chris Anderson refers to the "Long Tail" as a subset of the product space that contains niche goods and services outside of the mainstream. Yin et al. [21] argue that traditionally, the Pareto Rule, or the 80/20 rule, seemed to hold in that a large amount of company revenues was generated by relatively few products. However, the Internet changed that dynamic. Now customers have access to a vast array of niche products that brick-and-mortar companies could not afford to hold in inventory. Yin et al. go on to state that the long tail offers higher profits for companies than the head market for following two reasons:

- Economic principles drive the profitability of mainstream items down due to a high number of competitors being able to offer them. In contrast, long tail items can be sold at a higher profit margin.
- Offering long tail products creates a "one-stop" shopping experience [5], where customers are inclined to also purchase mainstream items, and thereby generating second-order or residual sales [21].

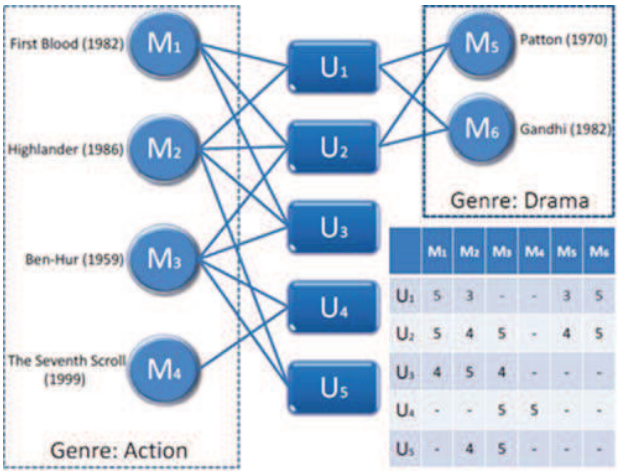
In support of the first point, we note that most head market products tend to be "vanilla" in nature in that their engineering is tempered to appeal to a wider audience. In contrast, niche products are specifically created to satisfy certain desires. As a result, their utility is higher. It follows that customers are more willing to pay high prices for these products.

1.2 Difficulty of Long Tail Recommendations

The head market can be represented as a dense matrix that lends itself well to collaborative filtering [4], matrix factorizations [9], nearest-neighbors [17], clustering [7], and traditional machine learning algorithms [20]. Matrices representing the long tail, on the other hand, are sparse. For example, Yin et al. [21] make the case that association rules, collaborative filtering, and matrix factorizations lead to local and obvious recommendations. In other words, these

methods lead to products that a given user would already be familiar with.

We first describe two novel methods proposed by Yin et al. [21] and Shang et al. [18] that enhance long tail recommendations. The former method modifies the probability variables of the hitting time algorithm to increase the likelihood that a random walker reaches the long tail regions of a bipartite graph, whereas the latter is a collaborative filtering method that employs a tripartite graph and random walkers to find users with similar tastes as a user u [1]. Our proposed method combines elements of these methods, as well as elements from the familiar PageRank algorithm, to increase the likelihood that u is recommended items (i) whose characteristics are appealing to u, and (ii) are found in the long tail region. Specifically, we reduce the Long Tail Recommendation problem to traversing a tripartite graph through a Markov process. In other words, we represent the tripartite graph as a stochastic matrix and after t (≥ 1) iterations make recommendations to u based on the probability of arriving at each of the items available.



	M1	M2	M3	M4	M5	M6
U1	5	3	-	-	3	5
U2	5	4	5	-	4	5
U3	4	5	4	-	-	-
U4	-	-	5	5	-	-
U5	-	4	5	-	-	-

Figure 1: Bipartite representation of user-item dataset

2 RELATED WORK

Recommendation systems have typically been divided into two categories: *content-based* filtering and *collaborative-based* filtering. As mentioned, collaborative-based filtering provides recommendations based on the users most similar to a user u or the products most similar to the products P rated by u [6, 12]. We mentioned earlier that the weakness of this process is that it tends to provide local, trivial recommendations. Content-based filtering refers to predicting a rating by u for a product p based on a feature vector of descriptions of products previously rated by u [16]. Unfortunately, most products lack elaborate descriptions, which makes for poor predictions [21]. In addition, products that do not have similar features to those rated by u are not recommended. In other words, this method provides no novel recommendations. The usage longevity of collaborative-based filtering and content-based filtering is that they tend to do well in the head market. However, as mentioned before, the real value to users comes from long tail recommendations, i.e., the introduction of products they would likely never discover through a non-algorithmic search.

Two major works propose algorithms for recommendations based on k-partite graphs. As previously mentioned, the usage of graph prevents the necessity of imputing values or needing a significant number of like users before effectively recommending. We show a detailed representation of each.

2.1 Hitting Time and Absorbing Cost

Yin et al. [21] represent a set of users U and a set of items M in a bipartite graph, which has a corresponding adjacency matrix. (Figure 1 shows a representation of how such a matrix is translated into a bipartite graph.)

In the hitting time algorithm in [21], the weights on the edges are given as

$$\begin{aligned} p_{i,j} &= P(s(t+1) = j \mid s(t) = i) \\ &= \frac{a(i,j)}{d_i} \end{aligned} \tag{1}$$

where $d_i = \sum_{j=1}^{n} a(i,j)$. The hitting time, or the number of steps it would take a user q to reach a product j while traversing this graph, is given by

$$\begin{aligned} H(q|j) &= \frac{1}{p_{j,q}} \\ &= \frac{\pi_j}{p_{q,j}\pi_q} \end{aligned} \tag{2}$$

where

$$\pi_i = \frac{\sum_{j=1}^{n} a(i,j)}{\sum_{i,j=1}^{n} a(i,j)} \tag{3}$$

The metric $H(q|j)$ has the unusual quality that a low hitting time means q and j are relevant and few users have rated item j. In other words, j is a product that is similar to those q has rated high and j is in the long tail. This ingenious approach maps q to items that are in the long tail and potentially appealing to q.

While this method takes into account only the relationship between users and ratings, Yin et al. merely use this method as a base case and enhance the method with other features. The primary algorithms proposed by Yin et al. are the Absorbing Cost (AC) algorithms of the form

$$AC(S|i) = \begin{cases} 0 & i \in S \\ \sum_j p_{ij} c(j|i) + \sum_i p_{ij} AC(S|j) & i \notin S \end{cases} \tag{4}$$

where p_{ij} is the *probability* of following a path from i to j, and $c(j|i)$ is the *transition cost* from a state i to its adjacent state j. While the hitting time algorithm just employed the ratings that users gave to items, the Absorbing Cost (AC) algorithm entails other features. For example, a user's distribution θ of item categories is used to define the cost $c(j|i)$. Hence, the proposed methods tap into data beyond simple user-movie ratings.

2.2 Tripartite Approach

Shang et al. [18] take a similar approach in that they also use a k-partite graph; however, there are three key differences between their method and Yin et al.'s approach.

(i) They employ a tripartite graph by additively combining the results of a user-item bipartite graph and a user-tag bipartite graph.

(ii) Ratings are not considered, rather a binary representation of (0, 1) corresponding to (connected, not connected) is employed.

(iii) The algorithm maps a user to a subset of users who are similar to a tag instead of mapping a user to items as in the hitting time algorithm.

Specifically, Shang et al.'s method goes as follows: for a user u and an object α that u has collected, set $a_{u\alpha} = 1$, and $a_{u\alpha} = 0$, otherwise. In the case of a tag s, $a'_{us} = 1$ if u has tagged an object with s, and $a'_{us} = 0$, otherwise. Given a target user v, Shang et al. first distribute the resource (or rating power, usually set to 1 initially) of v to

$$r_{\alpha v} = \frac{a_{v\alpha}}{k(v)} \tag{5}$$

where $k(v)$ is the *degree* of v in the user-item bipartite graph. Hereafter, the similarity between u and each user v with v being the target user is calculated as

$$\begin{aligned} s_{uv} &= \sum_{\alpha \in O} \frac{a_{u\alpha} \times r_{\alpha v}}{k(\alpha)} \\ &= \frac{1}{k(v)} \sum_{\alpha \in O} \frac{a_{u\alpha} \times a_{v\alpha}}{k(\alpha)} \end{aligned} \tag{6}$$

where $k(\alpha)$ is the *degree of item* α in the user-item bipartite graph, and O is the set of items.

The similarity measure between the tags of u and the tags of v is computed as

$$s'_{uv} = \frac{1}{k'(v)} \sum_{t \in T} \frac{a'_{ut} \times a'_{vt}}{k'(t)} \tag{7}$$

where $k'(t)$ ($k'(v)$, respectively) is the *degree* of tag t (user v, respectively) in the user-tag bipartite graph, and T is the set of tags. The two similarity measures are combined in additive fashion as follows:

$$s^*_{uv} = \lambda s_{uv} + (1 - \lambda) s'_{uv} \tag{8}$$

where λ is a tuning parameter. Once each s^* has been calculated for each (v, u) pair, the preference of v on α is computed as

$$p_{v\alpha} = \sum_{u \neq v} s^*_{uv} a_{u\alpha} \tag{9}$$

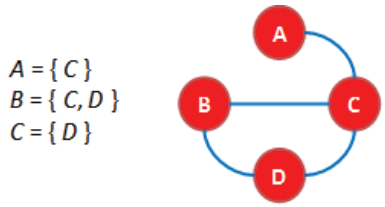


Figure 2: Directed graph of a simple system

All of the items that v has not rated are sorted and the top-n items are recommended to v. As was shown, this system is a direct *collaborative filtering* system, while the hitting time algorithm employs a *random walker* to connect a user v to items by traversing user nodes.

3 GRAPH TRAVERSAL THROUGH A MARKOV PROCESS

Before proposing our solution, we offer a brief explanation (through an example) of how a *Markov process* describes the *probabilities* of a random walker arriving at a node j from a node i, which we term $p_{i,j}$, given an increasing time horizon t.

EXAMPLE 1. Let $\{A, B, C, D\}$ be a set of nodes, its adjacency lists and the corresponding graph are as shown in Figure 2.

The system could be represented as a stochastic transition matrix L as follows:

$$L = \begin{pmatrix} 0.0 & 0.0 & 0.33 & 0.0 \\ 0.0 & 0.0 & 0.33 & 0.5 \\ 1.0 & 0.5 & 0.00 & 0.5 \\ 0.0 & 0.5 & 0.33 & 0.0 \end{pmatrix}$$

Hence, the initial probability of arriving at D from C is given by $a_{3,4}$ or $\frac{1}{3}$. Note that the columns of L sum to 1, i.e., $\sum_j p_{i,j} = 1$, for all time t. We can find the probability of $p_{i,j}$ for any t by simply calculating $l_{j,i} \in A^t$. Let's analyze the process of arriving at A from D. At time $t = 1$, we have

$$L^1 = \begin{pmatrix} 0.0 & 0.0 & 0.33 & 0.0 \\ 0.0 & 0.0 & 0.33 & 0.5 \\ 1.0 & 0.5 & 0.00 & 0.5 \\ 0.0 & 0.5 & 0.33 & 0.0 \end{pmatrix}$$

which is the *initial* representation and shows no path from D to A, whereas at $t = 2$, we have

$$L^2 = \begin{pmatrix} 0.33 & 0.16 & 0.00 & 0.16 \\ 0.33 & 0.42 & 0.16 & 0.16 \\ 0.00 & 0.25 & 0.66 & 0.25 \\ 0.33 & 0.16 & 0.16 & 0.42 \end{pmatrix}$$

Hence, there is a $\frac{1}{6}$ chance of arriving at A from D at $t = 2$. There are five possible 2-step trips that we could have taken from D, i.e.,

$$\{\{B, C\}, \{B, D\}, \{C, A\}, \{C, B\}, \{C, D\}\}$$

Of these, only one lands us in A. Note that our chance of arriving at C from D increases with respect to our chance of arriving at any other node as t increases. That is because C has the *highest* degree of the set of nodes in the graph.

As t increases, the elements of L converge to probabilities that do not change with subsequent iterations of the *Markov process*. This state is known as the *stationary distribution*. For this example, the stationary distribution is

$$L^{38} = \begin{pmatrix} 0.125 & 0.125 & 0.125 & 0.125 \\ 0.250 & 0.250 & 0.250 & 0.250 \\ 0.375 & 0.375 & 0.375 & 0.375 \\ 0.250 & 0.250 & 0.250 & 0.250 \end{pmatrix}$$

Therefore, for $n > 38$ the probabilities of L^n no longer change. Observe that for all nodes in $l \in L$, the most likely destination when leaving l is C, since the stationary distribution favors nodes with the *highest* degree, which should be kept in mind when we discuss our proposed solution. □

4 PROPOSED SOLUTION

The above-mentioned algorithms, as well as the Markov process, hold the critical elements in solving the Long Tail recommendation problem. The elements are specifically

(1) A third set of nodes that characterizes the set of users U and/or the set of items I to enhance the process.

(2) A Markov process allows us to simulate the hitting time algorithm by showing the probability of a random walker leaving a user node i and arriving at an item node j.

(3) We would like the process to favor items that pertain to genres *favorable* to a user, have *few* ratings, and have a *high* average rating.

We seek to combine these elements in our proposed solution, i.e., our method, as follows:

(1) Adds a set of nodes C that represents additional information relating I to U. This set of nodes creates additional paths from a user u to a preferred item i. We will show that if the information contained in the nodes of C adequately describe a relationship between I and U, the resulting tripartite graph will contain shorter paths between u and i than were available in the bipartite graph. In other words, there will be "express lanes" that more expeditiously connect u and i.

(2) Employs a Markov chain to simulate the hitting time algorithm. This allows us to quantify the probability of u arriving at each of the items $i \in I$.

Critical to our solution, however, is the balance of a trade-off between the implementations of our *tripartite graph* and the *Markov chain*. Specifically, while the tripartite graph adds a significant number of *shorter paths* between u and a preferred item i and the Markov process allows us to calculate the probabilities of u arriving at items that (s)he has not yet rated, as the time t for the Markov chain increases, the process will favor items with more links. In other words, as t increases so does the probability of arriving at an item with many ratings—a short-tail item. We give specific details of the implementation of the tripartite graph and Markov chain below and describe in more detail the benefit they provide our solution and the details of their implementations.

4.1 The Tripartite Graph

Representing our recommendation system as a tripartite graph has significant advantages over a simple bipartite graph. Consider the user-item bipartite graph as shown in Figure 3(a). The bipartite structure is such that user A would only be recommended items 1, 2, and 3 through a random walker. Since the long tail item 4 is disconnected from the user network of A, A will not be recommended this item. However, suppose we add another dimension to our dataset, such as genre. Let the set of genres be $\{S, T\}$ and the items belong to the following sets, $S = \{1, 4\}$ and $T = \{2, 3\}$. We now represent the dataset as a tripartite graph as shown in Figure 3(b). Now, A has a path to long tail items through the category S. By including extra dimensions to the graph, we allow users to access more items in the long tail that are to their liking, e.g., the relatively few links at item 4 would give the random walker a higher probability of being at item 4.

4.2 Connecting the Tripartite Graph

As mentioned earlier, the bipartite graph connects a set of user nodes U and item nodes I through a set of edges E, where an edge $(i, j) \in E$ connects a user i to an item j. The weight of the edge w is the *rating* that i assigned to j.

We introduce a third set of nodes C. The elements in the set are *features* that describe U or I.[1] Note that this approach differs significantly from that of Shang et al. [18], since the latter connect U to I and U to C separately and then additively combine the hitting times from each user u_i to another user u_j. Hence, their approach would represent the system shown in Figure 3(b) as in Figure 3(c) instead. Hence, a random walker would never be directly connected from I to C. The resultant connections from U to C and U to I essentially combine the results of two bipartite graphs, but does not directly join C into the bipartite graph, whereas our method allows for traversal between all sets I, U, and C, even though it comes at a cost.

4.3 Limitations of Tripartite Graphs

By employing the tripartite graph as shown in Figure 3(b), we lose the hitting time property of non-symmetry between $a_{i,j}$ and $a_{j,i}$. In short, this is the strength behind the hitting time algorithm and absorbing cost algorithms proposed by Yin et al. [21]. Their algorithms perform well because the number of steps taken from an item j to a user u better describes the uniqueness of the item than calculating the number of steps to reach j from u. Since we lose this property of non-symmetry, we are not able to have the Markov process mature to a *stationary distribution*. However, the process becomes valuable in its earlier stages (i.e., $t < 7$), since it allows us

[1] These features could contain demographic information C such as age, gender, or occupation. They could also describe the items, such as genre being a description we used earlier.

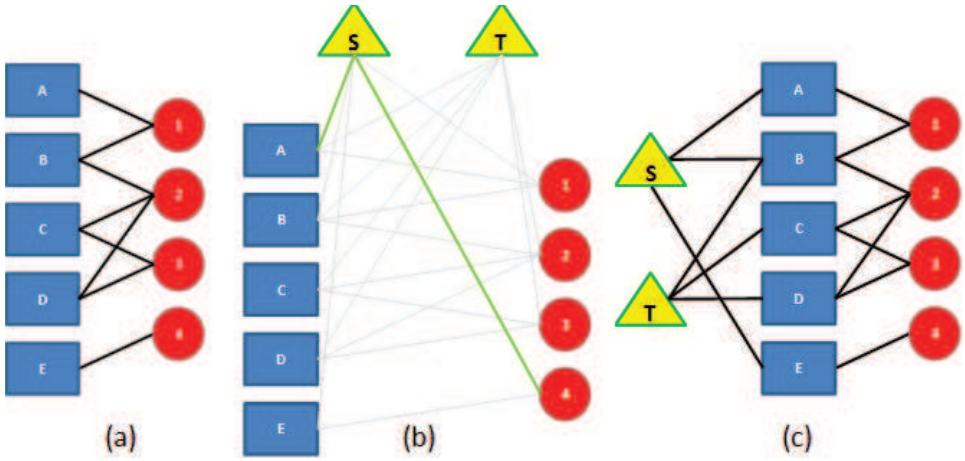


Figure 3: (a) A simple user-item bipartite graph, (b) a tripartite graph with new path created from A to 4, and (c) the Tripartite approach presented by [18]

to see how a random walker would traverse the graph given a short time horizon. We will argue that the short time horizon is a better representation of reality than the stationary distribution.

4.4 Traversing the Bipartite Graph Through a Markov Process

Initially, a tripartite graph as applied to the Long Tail Problem only shows connections between a user u and the items (s)he has rated and between u and the genre of items rated by u. Clearly, this is not useful in predicting ratings for items not yet rated by u. The Markov process, however, allows us to see the probabilities of arriving at different item nodes for each increasing value of t as demonstrated in Example 1. With respect to Example 1, the graph initially provides probabilities for

$$\begin{aligned} \frac{8}{|nodes|^2} &= \frac{8}{16} \\ &= 50\% \end{aligned}$$

of the possible paths. However, for $t = 2$, we have the probabilities for $\frac{14}{16} = 87.5\%$ of the possible paths. By $t = 4$, we have a probability for all possible paths. With respect to the MovieLens dataset[2] that we have analyzed, there are initially 1,000,209 ratings, which represents

$$\begin{aligned} \frac{1,000,209}{|users| \times |movies|} &= \frac{1,000,209}{6,040 \times 3,607} \\ &= 4.59\% \end{aligned}$$

of the total possible ratings. However, observe the following values for t:

$$t = 2 : \frac{13,515,389}{6,040 \times 3,607} = 62\%$$

$$t = 3 : \frac{22,384,240}{6,040 \times 3,706} = 100\%$$

Hence, each of the users $u \in U$ has a 100% probability of arriving at all the items $i \in I$ after $t = 2$.

4.5 Connectedness-Descriptiveness Trade-Off

So far we have described two fundamental elements to our proposed recommendation system. First, an additional set of nodes C that greatly enhances *descriptive* representation of the relationship between U and I and thus a potentially greater infrastructure for building short paths between U and the long tail items in I. Second, the *Markov chain* that allows us to generate probabilities for arriving at *unrated* items based on our new tripartite structure.

Assume that we can quantify the *descriptiveness* of the tripartite graph based on how the nodes are currently connected. In other words, we assign the degree to which the graph actually represents the tastes of the users in U with a value denoted D. Initially, D is high as it shows connections between the users and items they actually rated. This arbitrary value D can be greatly enhanced by adding the nodes in C in that we now know how users prefer different genres and how those genres related to the items in I. Now, let us quantify the *connectedness* of graph as an arbitrary value T. This value has a more concrete representation as it can be associated with the degree to which the Markov process has progressed towards a stationary distribution. Initially, this value is low. Figure 4 shows the abstract representation of how D and T relate as t increases.

The critical relationship is that as the Markov chain progresses towards a stationary distribution, the *connectivity* obviously increases. However, the *descriptiveness* is eroded in that the degree to which the process has progressed towards a stationary distribution is the degree to which the process merely favors nodes with a high degree,

[2] http://www.grouplens.org

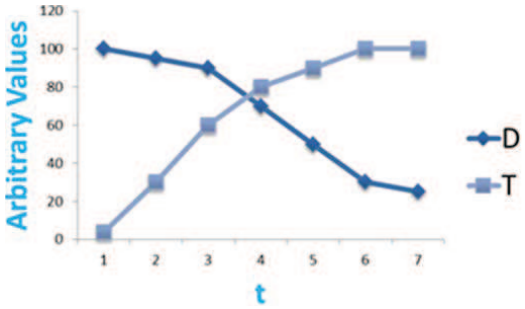


Figure 4: Abstract depiction of Connectivity-Descriptiveness trade-off as a function of t

or short head items. At that point, the paths no longer represent the potential infinite variations that we see in user tastes. They simply assign to a user u the items i that are most popular. We argue that the stationary distribution is not representative of a user's preference. As shown in Figure 4, the descriptiveness of the graph is *reduced* to returning for each node n,

$$\frac{\text{degree of} n}{|\text{nodes in } L|^2}. \tag{10}$$

For this reason, we employ the Markov process for small values of t in order to increase the *connectivity*, but end the process once the *descriptive* power of the graph significantly decreases, which lies the *Connectedness-Descriptiveness* trade-off.

5 IMPLEMENTATION DETAILS

Up till now, we have merely described principle foundation of our recommendation process. We now describe the details of the implementation.

5.1 Edge Weights

With respect to the edges connecting U and I, for a user u and item i the *edge weight* w is computed as follows:

$$w_{u,i} = \frac{rating(u,i)}{\sum_{j \in I} rating(u,j)} \tag{11}$$

where $rating(u,i)$ refers to the rating given to item i by user u.

Creating the set of edges between U and C and the set of edges between I and C, however, is not as straight forward. That is because the nodes of C generally have direct connection with either U or I, but not both. As we often deal mainly with the genre of an item as our set C in our experiments, we describe the approach to creating edge weights from C to U and C to I by illustrating the process of adding genres to our bipartite graph. Given that genres describe items, we would connect a genre $c \in C$ to an item $i \in I$ if i corresponded to c and set as its *weight* the average rating given to i. While we could simply set a binary weight representing either membership or no membership, the *average rating* facilitates random walkers to gravitate towards the better rated movies.

For the connecting weights between C and U, a genre c is connected to a user u if u has rated at least one item that corresponds to c. The *weight* of the connection is the average rating that u gave to the set of items (s)he rated that correspond to c. Hereafter, we create the edge weight connecting u to c as follows:

$$w_{u,c} = \frac{avg\_rating(u,c)}{\sum_{j \in C} avg\_rating(u,j)} \tag{12}$$

where $avg\_rating(u,c)$ is the average rating that u gave to items of feature c.

The edge weight connecting a feature c to an item i is defined below.

$$w_{c,i} = \frac{avg\_rating(i)}{\sum_{j \in C} avg\_rating(j)} \tag{13}$$

where $avg\_rating(i)$ is the average rating given to i from all users. Hence, when a random walker arrives at a feature c from a user u, the walker will have a greater probability of arriving at an item i that has been rated highly.

We note that the relationship of nodes from C to I, as defined in Section 6, is one to many, while the relationship between C to U is many to many. The former is important because items can be from several different genres. For example, a movie may be a romantic comedy. However, for the purposes in conducting our empirical study, we considered a romantic comedy to be a separate genre than a comedy or a romance. Whether this is the most effective method would be a rich area of future experimentation. Indeed, the methodology behind adding a third, or perhaps fourth or more, element to the bipartite graph should prove to be fruitful. However, it is likely that the complexity of the procedure is what has discouraged more use of k-partite graphs, where $k > 2$.

5.2 Adjacency Matrix

The tripartite graph is translated to an *adjacency matrix* as would be the case for representing any graph as a matrix. For example, the graph shown in Figure 3(b) could easily be translated to an adjacency matrix where we to have the ratings for $\{A, B, C, D, E\}$.

6 EXPERIMENTAL RESULTS

We used the MovieLens dataset[3] for conducting the performance evaluation of our proposed Long-Tail recommendation system and compare our method against those proposed by Yin et al. [21] on two metrics: *Recall@N* [10] and *Diversity* [3], which are formally defined below. These methods were used in [21]. We compared our experimental results to those in [21] using the MovieLens dataset and treated as our baseline the *hitting time algorithm* proposed in [21]. First of all, we divided the MovieLens dataset into a training set M and a testing set T. For each movie item i given 5 stars by a user u, we randomly selected 1,000 items not rated by u and compute the scores for i, as well as the 1,000 items. We then formed a ranked list r of these 1,001 rated items. Based on this list, $hit@N$ = 1 if $i \in r$, and 0 otherwise.

$$Recall@N = \frac{\sum hit@N}{|L|} \tag{14}$$

[3]The MovieLens database is a large movie ratings database composed of approximately 11 million ratings of around 8,500 movies.

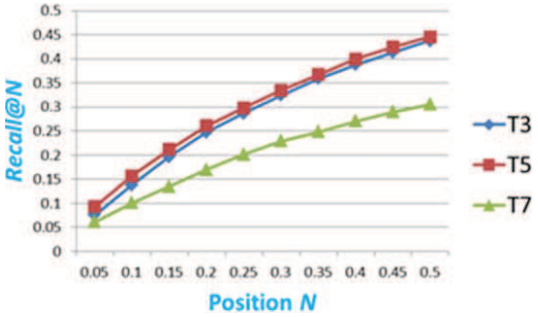


Figure 5: *Recall@N* **for** $t = \{3, 5, 7\}$

where $|L|$ is the size of the test set, and

$$Diversity = \frac{|\cup_{u \in U} R_u|}{|I|} \quad (15)$$

where U is the set of users, I is the set of items, and R_u is the set of recommendations for user $u \in U$.

We randomly selected 200 users and formed a set from the top-10 items recommended to these users. A high *diversity* score translates into more long-tail items being recommended [8]. The intuition is that if a method recommends relatively few items, these items tend to be popular short head items [15]. By contrast, the *greater* the number of unique items recommended to the set of users, the *greater* the probability that long tail items are found among such a set.

We ran our proposed solution for values of $t = \{3, 5, 7\}$ and compared them to the two absorbing time algorithms (AC1 and AC2) and the *hitting time* (HT) algorithm proposed by Yin et al [21].

6.1 Performance Evaluation

We first measured *Recall@N* for our proposed recommendation system based on T3, T5, and T7 referring to the Markov process where $t = 3$, 5, and 7, which is depicted in Figure 5. The results, as shown in Figure 5, illustrates that our recommendation algorithm works best for the lower values of t. As mentioned previously, as the process progresses towards the stationary probability, the descriptiveness of the graph is lost. Hence, for $t = 7$, the ability of the model to recommend long tail items breaks down. Figure 6, which shows the comparisons between our results and those archived by [21], indicates that our proposed recommendation system at both $t = 3$ and $t = 5$ edges out the absorbing cost methods for Position $N \leq 0.40$.

In comparing the performance of our recommender with the one achieved by [21] in terms of the *diversity* metric, Table 1 shows that the *hitting time* [13] and *absorbing cost* [11] algorithms recommend a wider diversity of items by a significant margin. However, we purport that the proposed solution recommends primarily *higher* rated items given that the paths that reach I from U passing through C arrives at nodes representing items in I that have a higher rating. For this reason, our approach favors *higher* rated items.

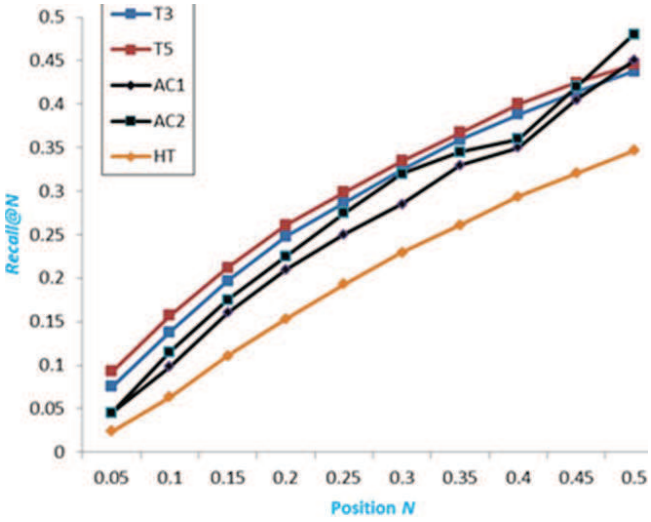


Figure 6: Results of our proposed method compared to the two absorbing cost methods and a hitting time method adopted by [21]

6.2 Performance Evaluation Summary

We purport that the proposed algorithm performs very well compared to the excellent methods proposed in [21] due to the fact that the added set of feature nodes C provide an *express lane* from a user u to the set of items I that are highly rated. The critical component to guiding a user to items that are specific to the user is that the number of nodes in C is 301, which represents the combination of genres that describes a movie, rather than the 18 atomic genres. This puts the random walker into a more specific category where it has a higher chance of reaching quality long-tail items through other users connected to those items. As depicted in Figure 7(a), a random walker arrives at a genre of one of the movies favored by u. After the walker arrives at the items rated highly that are from the feature c, the random walker can then connect back with users who have similar tastes (see Figure 7(b)). Furthermore, From these users, the random walker can arrive at items within the same space of I and thereby find potential highly rated long-tail items (see Figure 7(c)). Hence, the proposed algorithm targets items in specific regions of the graph rather than a "shotgun" approach of recommending a greater number of unique items across all users.

7 CONCLUSION

The seminal works of Yin et al. [21] and Seng et al. [18] have laid the foundation for representing the user-item interaction as a bipartite graph and appropriately traversing it in order to provide recommendations. We adopt elements from these methods to create a solution that employs a *tripartite graph* and a *Markov process*. This allows us to extract the most value from each of these tools in order to direct random walkers to regions of the graph that are specific to the users' tastes and thereby increase the chance that the random walker reaches a quality long-tail item.

To the best of our knowledge, our proposed long-tail recommendation system is the only system that consider using the tripartite

Table 1: Diversity scores

Algorithm	Diversity Score	Algorithm	Diversity Score	Algorithm	Diversity Score
t = 3	0.27	AC1	0.425	HT	0.41
t = 5	0.15	AC2	0.42		

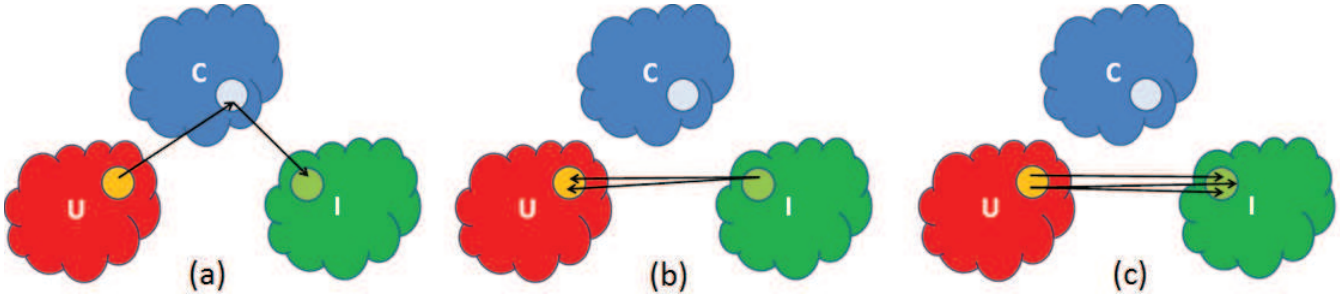


Figure 7: (a) The sets of nodes U, C, and I, where a random walker arrives at a set of nodes that represent movie genres rated high by a user u, (b) Connecting back to users with similar preferences for items that pertain to feature c, and (c) Random walker directed to highly rated items that fall under c

graph and a Markov process to suggest long-tail items to users. The performance evaluation as presented in the Experimental Results section verifies the *uniqueness* and *novelty* of our recommendation approach, which further enhances existing algorithms in making long-tail item recommendations.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin, *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*, IEEE TKDE, Vol. 17, No. 6, pp. 734-749, June 2005.

[2] C. Anderson. The Long Tail: Why the Future of Business is Selling Less of More, Hachette Books, 2008.

[3] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Buttcher, and I. MacKinnon, "Novelty and Diversity in Information Retrieval Evaluation", ACM SIGIR, pp. 659-666, 2008.

[4] M. Ekstrand, J. Riedl, and J. Konstan, *Collaborative Filtering Recommender Systems*, Foundations and Trends in Human Computer Interaction, Vol. 4, No. 2, pp. 81-173, 2011.

[5] S. Goel, A. Broder, E. Gabrilovich, and B. Pang. "Anatomy of the Long Tail: Ordinary People with Extraordinary Tastes", ACM WSDM, pp. 201-210, 2010.

[6] J. Herlocker, J. Konstan, and J. Riedl. *An Empirical Analysis of Design Choices in Neighborhood-based Collaborative Filtering Algorithms*, Information Retrieval, 5:287-310, October 2002.

[7] K. Kima and H. Ahn, *A Recommender System Using GA K-means Clustering in an Online Shopping Market*, Expert Systems with Applications, Vol. 34, No. 2, pp. 1200-1209, February 2008.

[8] S. Kingrani, M. Levene, and D. Zhang, "Diversity Analysis of Web Search Results", Article No. 43, ACM WebSci, 2015.

[9] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems", IEEE Computer, pp. 42-49, August 2009.

[10] C. Manning, P. Raghavan, and H. Schütze. Introduction to Information Retrieval, Cambridge, 2008.

[11] C. Mavroforakis, M. Mathioudakis, and A. Gionis, "Absorbing Random-Walk Centrality: Theory and Algorithms", IEEE ICDM, pp. 901-906, 2015.

[12] M. McLaughlin and J. Herlocker, "A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience," ACM SIGIR, pp. 329-336, 2004.

[13] Q. Mei, D. Zhou, and K. Church, "Query Suggestion Using Hitting Time", ACM CIKM, pp. 469-478, 2008.

[14] Y. Park and A. Tuzhilin, "The Long Tail of Recommender Systems and How to Leverage It", ACM RecSys, pp. 11-18, 2008.

[15] R. Reinanda, E. Meij, and M. de Rijke, "Document Filtering for Long-tail Entities", ACM CIKM, pp. 771-780, 2016.

[16] F. Ricci, L. Rokach, B. Shapira, and P. Kantor. Recommender Systems Handbook, Springer, New York, 2011.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms", WWW, pp. 285-295, 2001.

[18] M. Shang, Z. Zhang, T. Zhou, and Y. Zhang, "Collaborative Filtering with Diffusion-based Similarity on Tripartite Graphs", Physica A: Statistical Mechanics & Its Applications, Vol. 389, No. 6, pp. 1259-1264, 2010.

[19] D. Shen, X. Wu, and A. Bolivar, "Rare Item Detection in E-Commerce Site", WWW, pp. 1099-1100, 2009.

[20] G. Webb, M. Pazzani, and D. Billsus, *Machine Learning for User Modeling*, User Modeling and User-Adapted Interaction, Vol. 11, No. 1, pp. 19-29, March 2001.

[21] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, "Challenging the Long Tail Recommendation", VLDB Endowment, Vol. 5, No. 9, pp. 896-907, 2012.