On the Long Tail Products Recommendation using Tripartite Graph

Arlisa Yuliawati, Hamim Tohari, Rahmad Mahendra, Indra Budi
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia

***Abstract*—The growth of the number of e-commerce users and the items being sold become both opportunities and challenges for e-commerce marketplaces. As the existence of the long-tail phenomenon, the marketplaces need to pay attention to the high number of rarely sold items. The failure to sell these products would be a threat for some B2C e-commerce companies that apply a non-consignment sale system because the products cannot be returned to the manufacturer. Thus, it is important for the marketplace to boost the promotion of long-tail products. The objective of this study is to adapt the graph-based technique to build the recommendation system for long-tail products. The set of products, customers, and categories are represented as nodes in the tripartite graph. The *Absorbing Time* and *Hitting Time* algorithms are employed together with the *Markov Random Walker* to traverse the nodes in the graph. We find that using *Absorbing Time* achieves better accuracy than the *Hitting Time* for recommending long-tail products.**

Keywords*—Long tail; recommender system; tripartite graph; random walker; hitting time; absorbing time

I. Introduction

Promotion becomes one of the success factors in product marketing [1], i.e the better the promotion, the more people recognize the products being promoted, and the higher chance for those products being sold. On the other hand, the wrong strategy in promoting products could also cause difficulty or even failure in selling specific products [2]. In general, e-commerce companies tend to recommend popular products to customers. Those products would remain popular and the not recommended products would be less exposed by customers. The high number of unpopular products would be harmful to some B2C (Business-to-Consumer) e-commerce companies that apply the non-consignment sale system. These e-commerce companies have already paid the products from the manufacturer to be sold to the customers. The products will not be returned even though the companies are unable to sell them. The more unpopular products are unsold, the higher the cost would be due to the damage risk or the inventory cost for storing such products.

The long-tail phenomenon is a condition when the unpopular (niche) products dominate the total sales [3]. Long-tail products are also interpreted as the less popular products among customers [4]. Even though the sales volume of each product was not so high, the total number of products dominate the total sales [2]. The ratio between long-tail products and popular products is following the 80/20 principle or Pareto rules. The 80% of total revenue is obtained from 20% of total products, i.e., the popular ones. By increasing the sales volume of the remaining 80% of the total products (the long tail), the total revenue could be increased significantly.

A recommendation system is one of the important tools for marketing strategy. It is useful in dealing with the information overload issue as the variation of the products increases. Many studies in the recommendation systems for the e-commerce domain have been conducted [5], [6], [7], [8], [9]. Studies in this area are usually focused on the behavior or characteristics of the ”known” products or ”the shopping history” of customers. The common objective is to recommend the most suitable products based on transaction history. By its ability to capture customers' preferences, it is easier to recommend such suitable products for them. And for the customers, it will be easier to determine which to purchase and where to buy. On the other hand, since the recommendation commonly brings popular products up, these products become more competitive among many business owners [4]. Thus gaining profit from such products could be more challenging. On contrary, the less popular products are less noticed by many sellers so they could bring more profit if it is successfully sold [10], [4].

The characteristic of the data being used in a recommendation system is suitable to be represented in a graph. In every domain of the recommendation system, it is possible to represent the entities, such as users, items, movies, foods, images, books, as the nodes (vertices) of a graph. Meanwhile, any relations between entities can be represented as the edges. Many studies about graph-based recommendation systems have been conducted for different problems to be solved. A graph structure was owned by mostly recommendation system and also it raises many potential exploration and development through graph learning [11]. One common graph representation is a bipartite graph, for example, to capture the relationship between a set of users and a set of items. A study by [12] used this kind of representation to apply collaborative filtering based on user similarity and item similarity. A bipartite graph was also used by [2] to solve the long-tail problem through a random walker that is adopted in this study. The other example with different graph representation but also employed the random walker is a study by [13]. It solves the cold start problem through a trust network by applying trust-based and item-based collaborative filtering.

Now recalling the non-consignment sale system applied to some B2C e-commerce companies, besides capturing customers' preferences, it is also important to take the unpopular products out to the customers. The motivation behind this study is to find the recommended products, which not only focus on the more popular products but also those which are less exposed by customers but still in the customers' preference area. Adopting previous studies, a tripartite graph representation is used to draw the relation between users, items, and categories. Since customers nodes are only connected to product nodes that they have ever purchased, then to make them exposed by the long-tail products, the *Markov Random Walker* combined with *Hitting Time* or *Absorbing Time* is employed to find the unpopular yet suitable products to be recommended to the users. In addition, as the product categories are available at different levels, this study also tries to figure out whether the different category level being used affects the recommendation results. More specifically, studies about the recommendation system for long-tail products are presented in Section 2 followed by the detail about tripartite graph implementation in Section 3. Section 4 consists of the experimental result and its analysis, and the conclusion would be presented in the last section.

II. Related Work

Studies to deal with the long-tail problem have been widely conducted. A study in [14] tries to analyze and solve the long-tail problems on the traditional recommendation system (i.e. collaborative filtering) on an e-commerce platform. By capturing the users' information and behavior together with the systems' behavior, several models are established involving Gradient Boosting Decision Tree, Logistic Regression, and user entropy-based LDA. This study shows that it is possible to recommend long-tail products while maintaining the quality of the recommendation.

A similar conclusion was also obtained from a study in [15] to enhance the collaborative filtering such that it considers mining the long-tail items in the recommendation process. This study was conducted on the sales of alcoholic beverages (RateBeer). A matrix factorization was established based on personal experience to generate the user experience level. The top N recommendation is then obtained from the experience level together with the consideration of items' popularity. This study captured a phenomenon where the customers with lower experience levels tend to purchase popular products more, and vice versa: those with higher experience levels, tend to purchase the unpopular ones. The problem of this study happened for new customers. The recommendation was either not relevant or only focused on the popular products.

Specific to deal with the cold-start problem and long-tail problem, a study by using social data (Flickr, BlogCatalog, YouTube, HetRec11-LastFM) conducted in [10] first decompose the overall products into the low-rank (short-head) products and the sparse part (long-tail) products. These two groups were trained independently and the final recommendation from each group was merged then became the recommended products for the new users. But basically, this study focuses more on resolving the cold start problem while 'introducing' some items from the long-tail category in the recommendation. Experimenting on a similar domain, especially related to movie viewers data (through MovieLens and Last.fm), a study by [16] developed CORE (Cosine Pattern-based Recommender). This system allows product recommendation based on either the popular products (based on those which have been rated by a user) or the niche products (based on the *cosine* pattern. This study reported that the accuracy of the recommendation will be decreased when comes to dense data.

Other studies in recommendation system studies employ the graph representation. Specific for the long-tail problem in a movie data set, a study in [2] initiate the use of a bipartite graph to represent the user-item relation. *Markov Random Walker* was implemented to calculate the *Hitting Time*, *Absorbing Time*, and *Absorbing Cost* which were used to determine the ranking of the product recommendations. The performance of *Absorbing Cost* outperformed the other two on various measurement metrics used due to the characteristic of the *Absorbing Cost* that considers customer interests/preferences when giving a recommendation. This is suitable if all products are similar as in movie data because it is easier to recommend a movie to customers who have a specific interest in a particular genre compared to customers who have an interest in several genres.

Adopting the bipartite graph approach in [2], a study in [17] added latent information (i.e genre node) as a link between the customer node and the recommended product node such that the graph representation became a tripartite graph. By this improvement, it is possible to traverse from a user node x to the item node y that is not directly connected through the intermediate genre node z that might be indirectly connected to y (for example through the intermediate node). The result shows its ability to 'pick' the recommended items from the region that is suitable to the users' taste. A study in [4] makes an improvement in determining the latent information by using a single category. This study was also proposed a new approach to calculate the weight between product and category nodes to avoid the misleading caused by the use of direct average rating, namely the Bayesian averages. It shows the recall and the diversity score improvement compared to the former study in [17]. In both of these studies, the *Hitting Time* and *Absorbing Cost* was employed based on their performance in the study by [2].

Compared to the movie data (MovieLens) used by [17] and [4], the domain of this study (e-commerce data from B2C company) owns similar components. Both data own set of users, items (i.e. movies vs products), and category (i.e. genres vs products' categories) such that it is possible to build the tripartite graph representation. This study then adopts the approach used by [17] and [4] to build the tripartite graph-based recommendation system that employs the *random walker* to promote the long-tail item to the user. However, the characteristic of e-commerce data is different from the movie data, so this work differs from the former at some points. The first one is when customers often purchase products from certain categories than the other, it does not always imply that the category is preferable to the other. For example, because a customer often purchases snacks on e-commerce, this does not imply that the customer does not interested to purchase clothes or electronic devices. This is probably the customer prefers to buy clothes or electronic devices on offline stores rather than from e-commerce. This is surely different from movie data where the preference to watch movies from a specific genre generally implies the preference to the respective genre. From this condition, having information about what a customer often purchased from e-commerce is not too useful for the recommendation process, so that *Absorbing Cost* that works by considering such information to make a recommendation is not suitable to be applied on e-commerce data. Thus, even though [2] mentioned that *Absorbing Cost* could recommend better than *Hitting Time* and *Absorbing Time*, this study try the other way –to not include the *Absorbing Cost*. The second point is that latent information in the movie data (genre) is very different from the latent information in e-commerce data (category). The main difference is that movie genres have the same level, while the product categories are divided into general categories (top-level) to the most specific category (leaf-level). Since the relation (as well as the weight) between items and different levels of the category might be different, the use of different levels of product categories could be one thing to be elaborated on, whether or not it affects the result of the recommendations.

III. Tripartite Graph Recommendation System

A. Graph Representation

This study employs a tripartite graph, i.e. a graph $G = \{V, E\}$ which its node set V is partitioned into three disjoint node subsets V_1, V_2, and V_3 such that $V = V_1 \cup V_2 \cup V_3$ and for each $(u, v) \in E$, if $u \in V_i$ and $v \in V_j$, then $i \neq j$ [18]. The graph representation utilized in this study would be an in-directed graph since a relationship between two nodes implies the reverse relationship. There are three types of nodes such as the *user* nodes representing the customers, *item* nodes representing the products, and *category* nodes representing the categories. The relationship between nodes is represented as weighted edges. There are also three types of edges connecting nodes from different types, they are the *user-item* edges, *item-category* edges, and the *user-category* edges.

Illustration of a tripartite graph representation is presented in Fig. 1. The blue, red, and green nodes represent the group of user or customer nodes, category nodes, and the item or product nodes, respectively. The label for user nodes was taken from the customers' username, while the product category label represents the product category name, and the item nodes use the brand name of the products as their label. In Fig. 1 each edge connecting nodes in the different groups represent different relationships. For example, the user node labeled "agustini24" has a pair of edges with opposite directions that are connected to the "Home Living" node. These edges represent that "agustini24" purchased products belonging to the "Home Living" category, and vice versa, the products belonging to the "Home Living" category were purchased by a user with the username "agustini24". Similar relations are also applied for the other edges connecting nodes from different groups of nodes. The rest of this subsection discusses detailed information about edge representation.

1) User-Item Edges: These edges connecting the *user* nodes with the *item* nodes. The weight of this edge is 1 if a customer has given a rating to a product, and 0 otherwise. To avoid the density of the graph, edges whose weight is 0 are removed.

2) Item-Category Edges: These edges connect the *item* nodes to *category* nodes. This type of edge uses the average rating value from all customers for a product in a specific category as the weight. It is computed from the average rating for a product i (denoted by i_{avg}) divided by the total number

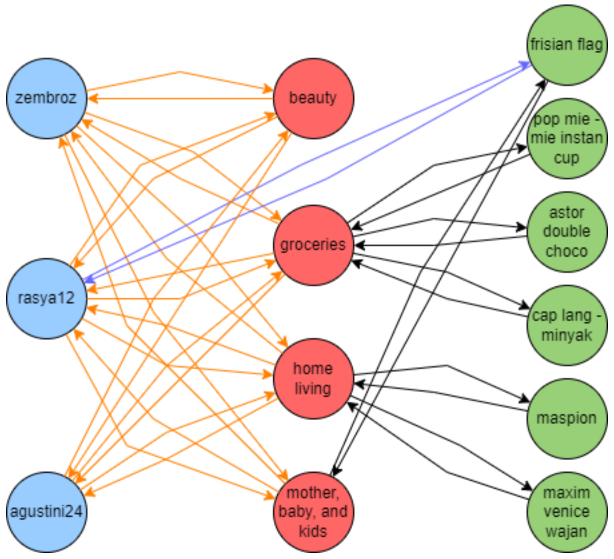

Fig. 1. An Illustration of the Tripartite Graph.

of categories connected to the product i (denoted by $|C|$). The formula is presented as

$$w_{i,cat} = \frac{i_{avg}}{|C|} \tag{1}$$

3) User-Category Edges: These last group of edges exist between the *user* and *category nodes*. These node connect *user* node to more *item* node with shorter paths. The weight is computed by using *Bayesian Average*

$$w_{u,cat} = \frac{avg\_votes_u \times avg\_rating_u + votes_{u,cat} \times rating_{u,cat}}{avg\_votes_u + votes_{u,cat}} \tag{2}$$

for edges connecting *user* node u and *category* node cat. The value of avg_votes_u is obtained by calculating the number of products bought by customer u in category cat divided by the total number of categories. The value of avg_rating_u is calculated by dividing the total ratings from customer u for all products with the total number of categories. The $votes_{u,cat}$ denotes the number of products in category cat that has been purchased by customers u. And the last, $rating_{u,cat}$ denotes the average ratings from customer u in a category.

B. Product Recommendation

This section contains an explanation about how to get the recommended long-tail product for certain customers by traversing the tripartite graph. The long-tail products are determined based on the average number of customers that give ratings to the whole product in the data set. A product is then labeled as long-tail if the number of customers that give a rating to it is below the average.

1) Markov Random Walker: A *random walk* is formed from a graph traversal such that given a starting node a, we choose an adjacent node b to be visited at random (usually based on predefined transition probability), then choose the next random node c to be visited from b and so on until certain steps [19], [11]. On a weighted graph, it forms a Markov Chain with the transition matrix consisting of the probability value of the movements between node i and j such that

$$p_{i,j} = \frac{w_{i,j}}{d_i} \tag{3}$$

where $w_{i,j}$ denotes the weight between node i to node j while d_i is the total weight of node i to all of its adjacent nodes [20]. In this study, $p_{i,j}$ refers to the probability of the *random walker* arrives at a product node j from a customer node i as the time t increases. The transition matrix (which is then denoted as M_S) is also called as a *stochastic matrix* since the sums of each row equals to 1.

In the matrix representation of a graph, the dot product of a matrix by itself for n times results in the availability of paths with length n between each pair of nodes. Related to this study, from a given *user* node, an iterative process is done by the *random walker* to find the suitable long-tail *item* node. This process is equivalent to the dot product of the *stochastic matrix* M_S by itself for t times which represents a *random walker* probability traverse from node i to node j in time $t (\geq 1)$. As the time t increases, the elements in this *stochastic matrix* converge such that no change in their value or the changes are very small. However, according to [17], it is better to use a small value of t since as it grows higher, the *random walker* tends to visit the popular nodes. Thus in this study, $t = 2, 3, ..., 7$ are used since after $t = 7$, the ability of the system to recommend the long-tail products was decreased [4].

2) Hitting Time: As stated in [2], *Hitting Time* (denoted as $H(q|j)$) is defined as the expected number of steps that is needed by a *random walker* to move from an *item* node j to *user* query node q with $j \neq q$. The value of *Hitting Time* is obtained from

$$H(q|j) = \frac{\pi_j}{p_{q,j} \cdot \pi_q} \tag{4}$$

where π_j and π_q are the *stationary probability* for node j and node q respectively. Meanwhile, $p_{q,j}$ represents the weight of edge connecting node q and node j, i.e. the movement probability between node q and node j. The smaller the value of $H(q|j)$ denotes the more relevant node q and j and that only a few users have rated item j. This conclusion comes from the following information.

- Consider the fact the value of *stationary probability* stays constant for all nodes, the value of the *Hitting Time* is inversely proportional to $p_{q,j}$. This means that the higher the value of $p_{q,j}$ which denotes the more relevant node q and j, then the lower the *Hitting Time* value will be obtained.
- The *stationary probability* of a node is proportional to the number of customers that give a rating to the product. This means that the lower the *stationary probability* of a product, then it belongs to the long-tail product because it is only rated by a few customers.

Algorithm 1. Recommendation by using Hitting Time

```
Input:
A tripartite graph $G = (V, E)$
A customer node $q \in V$
Time $t$ for how long the random walker traverse the nodes

Recomendation_By_HT($G, q, t$):
 1)  define a subgraph $G' = (V', E')$
 2)  for each node $j \in V$ that has not been rated by customer $q$:
 3)      include node $j$ as the member of $V'$ in $G'$
 4)      include edge $(q, j) \in E$ as the member of $E'$ in $G'$
 5)  create a stochastic transition matrix $M_S$ from $G'$
 6)  for each node $j$ in subgraph $G'$:
 7)      calculate the stationary probability $(\pi_j)$
 8)  perform dot products: $(M_S)^t$ represents the random walk length $t$
 9)  for each node $j$ in subgraph $G'$:
10)      calculate the Hitting Time value $H(q|j) = \frac{\pi_j}{p_{q,j}\pi_q}$
11)  sort the Hitting Time value for all node $j$ in ascending order, except
     node $q$
```

Algorithm 1 presents the steps to recommend the unpopular products (i.e. the long-tail products) based on the *Hitting Time* value. This algorithm intuitively tries to find the product nodes which has never been purchased by a customer but have higher similarity to those that have been purchased by the customer.

3) Absorbing Time: As the comparison of *Hitting Time*, the *Absorbing Time* is implemented regarding the study by [2]. It explains that *Absorbing Time* is suitable for data in which the number of customer nodes is far higher than the product nodes. Within this condition, the number of average rating for each product is higher than the number of average rating for each customer. Thus this information should be more useful for the recommendation process. *Absorbing Time* that is denoted by $AT(S|i)$ is defined as the expected number of steps before a *random walker* that is started from node i is absorbed by S. While the set S denotes the *Absorbing Nodes*, i.e. set of nodes $S \subseteq V$ in a graph $G = (V, E)$ for which the *random walker* stops when any node in S is reached for the first time.

$$AT(S|i) = \begin{cases} 0 & , i \in S \\ 1 + \Sigma_{j=1}^{n} p_{i,j} \cdot AT(S|j) & , i \notin S \end{cases} \tag{5}$$

calculates the value of *Absorbing Time*. It can be seen that $AT(S|i)$ would be 0 whenever the current node is one of the *Absorbing Node*, i.e those which are directly connected to the customer node. While a recursive calculation is performed from the source node (the customer node) to one of the *Absorbing Node*. This approach is similar to the one that uses *Hitting Time*, to find the less popular products (i.e. those belonging to the long-tail) and recommend products that are similar to what a customer has already purchased and rated. The difference is in its traversal route. By using *Absorbing Time*, the *random walker* traverses through the unpopular nodes until it arrives at a node that represents the popular one. The detail of the recommendation process through *Absorbing Time* is presented in Algorithm 2.

IV. Data and Evaluation

A. Data Set

This study adopts the approaches from previous studies [14], [17] by employing the tripartite graph representation to build the recommendation model for a B2C e-commerce

Algorithm 2. Recommendation by using Absorbing Time

```
Input:
A tripartite graph $G = (V, E)$
A customer node $q \in V$
Time $t$ for how long the random walker traverse the nodes

Recommendation_By_AT($G, q, t$):
 1)  define a subgraph $G' = (V', E')$
 2)  define $S \subseteq V'$ and $S' \subseteq V'$ such that $V' = S \cup S'$
 3)  for each product $i$ that has been rated by customer $q$:
 4)      include node $i$ as the member of $S$
 5)      include edge $(q, i) \in E$ as the member of $E'$
 6)  for each product $j$ that has not been rated by customer $q$:
 7)      include node $j$ as the member of $S'$
 8)  create a stochastic transition matrix $M_S$ from $G' = (S \cup S', E')$
 9)  perform dot products $(M_S)^t$ represents the random walk length $t$
10)  for each node $i$ in subgraph $G'$, calculate the Absorbing Time value:
11)      if node $i \in S$ then:
12)          $AT(S|i) = 0$
13)      else:
14)          $AT(S|i) = 1 + \Sigma_{j=1}^{n} p_{i,j} \cdot AT(S|j)$
15)  sort the Absorbing Time value in ascending order for all node in $S'$
```

TABLE I. Graph Data Set Description

The Number of	1st-level Category	3rd-level Category
Total nodes	63,068	3,011,244
Category nodes	21	626
Product/*item* nodes	20,000	20,000
Customer/*user* nodes	43,047	43,047
user-item edges	80,000	80,000
item-category edges	469,036	1,123,096
user-category edges	1,808,148	1,808,148
Average degree of each node	37	47

company. One of the differences is that in this domain, the product category has several levels. Thus, there are two types of data set being used in this study, differentiated based on the level of the category as summarized in Table I. There are in total 63,068 nodes that are connected to the 1-st level category nodes and 3,011,244 nodes connected to the 3-rd level category nodes. The objective of this differentiation is to identify the effect of the specialization (by using the 3rd-level category) and generalization (by using the 1st-level category) in the latent information for the recommendation result.

B. Evaluation Metrics

This study uses three metrics to evaluate the performance of the recommendation system adopting from [2]. These metrics evolved the evaluation of the accuracy, diversity, and exposure of the long-tail products. For each of these criteria, the comparison of *Hitting Time* and *Absorbing Time* performance are evaluated.

1) Evaluation on Accuracy: This metric use *Recall@N* that measures the accuracy of the recommendation result for each algorithm (*Hitting Time* and *Absorbing Time*). It evaluates how far the algorithm could recommend the long-tail products.

Given a collection of products consisting of the combination of customers' favorite products and other randomly chosen products, the recommendation system would recommend top N recommendations. If a customers' favorite product is included in the top N recommendation, the value of *hit@N* would be 1 and 0 otherwise. The notation $\left|L\right|$ represents the number of tests case, i.e. the total instances of long-tail products that are tested their membership to the top N recommendations. The formula for this metric is given in Equation (6).

$$Recall@N = \frac{\sum hit@N}{|L|} \tag{6}$$

2) Evaluation on Products Diversity: The purpose of this measurement is to identify the recommendation performance in term of a variety of products, whether the recommendation covers both the popular and unpopular products or only focus on the popular ones. The higher the diversity value, the more different types of products would be recommended to customers.

The value of *Diversity* is obtained from the comparison of the number of unique products being recommended by the system and the maximum amount of the recommendation. The amount of the recommendation is calculated from the multiplication of the desired number of top recommendations with the number of customers involved in the experiment. The formula for *Diversity* score is presented in Equation (7).

$$Diversity = \frac{|\bigcup_{u \in U} R_u|}{|I|} \tag{7}$$

3) Evaluation on Long-Tail Products: The last evaluation utilizes the *Long Tail* measurement. This metric determines whether the recommendation system successfully recommends long-tail products. Rather than using the average rating as conducted in [2] which could lead to the misleading implication, this metric is modified by considering the average number of customers who give a rating of a product. For example, product A is purchased by five people and all of them give 1 rating. Product B is only purchased by one person and the given rating is 5. If the *Long Tail* score is calculated using the average value, then product A will be considered as a long-tail product while product B is the popular product. Equation 8 presents the equation for calculating the *Long Tail* score. The notation $\left|L\right|$ denotes the number of tests, while *rating(i)* denotes the number of customers who give a rating on product-*i*.

$$LongTail = \frac{\Sigma_{i=1}^{n} rating(i)}{|L|} \tag{8}$$

V. Result and Analysis

Starting by entering a username of a customer into the system, a set of N products are generated as the recommended products for the customer, following Algorithm 1 (by using *Hitting Time*) and Algorithm 2 (by using *Absorbing Time*) separately. Each Algorithm is run by using two types of data set described in Subsection IV-A combined with a specific value of N (the number of products to be recommended) and t (the length of time needed by the *random walker*). The evaluation is conducted through the observation of the accuracy, diversity, and long-tail measurement.

A. Evaluation on Accuracy

The aim of this experiment is to identify the accuracy of the utilization of *Hitting Time* and *Absorbing Time* in different types of category levels, i.e the use of 1-st level category vs 3-rd level category. In general, the value of *Recall@N* is increasing with the increase of the number of products (N),

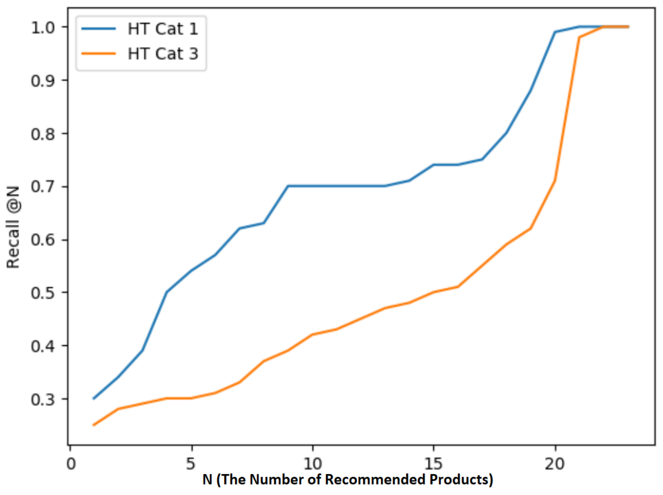

Fig. 2. The Accuracy (*Recall@N*) Value of using the Hitting Time.

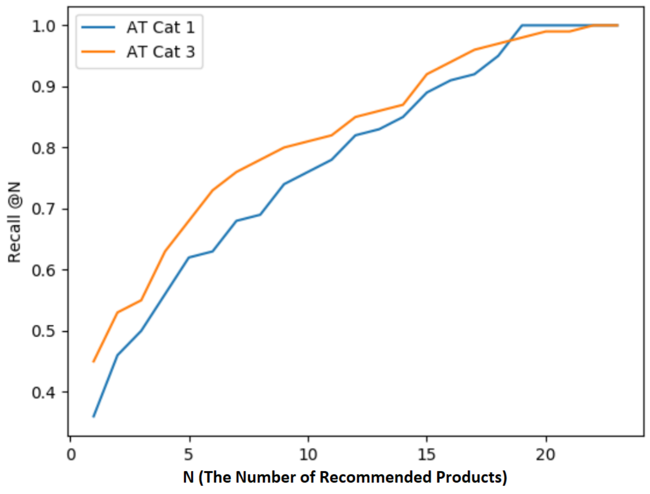

Fig. 3. The Accuracy (*Recall@N*) of using the Absorbing Time.

but different combinations of the algorithm and category level being used lead to different results.

Fig. 2 present the accuracy comparison by using different level of category in the implementation of *Hitting Time*. This result shows that the accuracy of using the general category (1st-level category) is better than the specific one. This is caused by the condition that the 1st-level category has a smaller number of *user-category* edges. This situation produces a higher rating average on each of these edges that drives to the higher probability value in the transition matrix. Thus, the *random walker* traverses the graph faster to reach the nodes around the customer query node, especially when the nodes are in the same category.

Fig. 3 presents the comparison of accuracy in different category levels by using *Absorbing Time*. If the use of *Hitting Time* resulting better accuracy when it is combined with the use of data from the 1st-level category, *Absorbing Time* performs better in its combination with the data from the 3rd-level category. This difference is affected by the working principle of both algorithms in determining the source and target nodes to be traversed by the *random walk* and the different characteristics of the connectivity in each level of category.

Regardless of the level of category data, both results presented in Fig. 2 and 3 shows that the curve of the *Absorbing Time* is higher than the *Hitting Time*. This implies that the implementation of *Absorbing Time* has a better performance in terms of its accuracy. Since *Absorbing Time* would run better in a graph with a shorter path between nodes, this condition is consistent with the fact that the data set consisting more customer nodes than the product nodes. Therefore, the connectivity between product nodes has a shorter path.

B. Evaluation on Diversity

In term of diversity, Fig. 4 shows the diversity comparison in top N recommendation ($N = 5, 10, 15, 20$). This figure shows that the implementation of *Absorbing Time* tends to

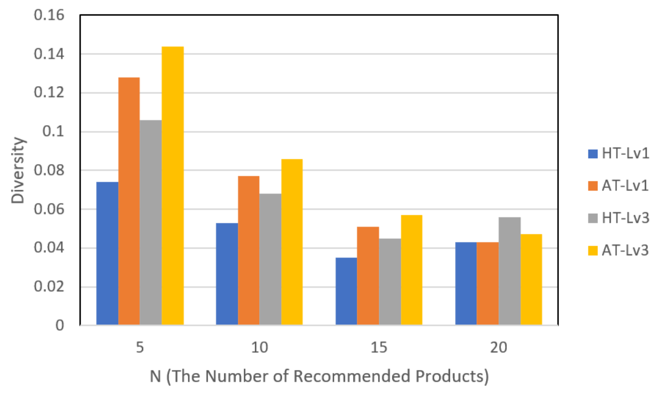

Fig. 4. The Comparison of Diversity Value by using *Hitting Time* and *Absorbing Time.*

be better than the *Hitting Time* in the small number of N. Moreover, the use of specific category data (i.e. 3rd-level) yields better diversity in both algorithms. From Fig. 4, it can be said that the greater value of N, the lower the diversity value would be obtained (for both approaches). However, this diversity value does not guarantee the quality (accuracy) of the recommendation because the accuracy is inversely proportional to diversity. Moreover, as summarized in [17] that the higher diversity value reflects the high probability of the long-tail products to appear in the recommendation. Thus, the *Hitting Time* algorithm produces the more diverse products, and probably captures the long-tails better than *Absorbing Time* algorithm, but might not be better in terms of accuracy.

TABLE II. THE LONG TAIL COMPARISON BY USING *Hitting Time* AND *Absorbing Time*

Top N Recommendation	*Hitting Time*	*Absorbing Time*
5	362.01	347.13
10	366.18	333.19
15	330.87	330.57
20	327.01	322.01

C. Evaluation on Long Tail

This evaluation calculates the average number of customers who give a rate to a product in each of the top recommendation levels. The lower average value denotes the better long-tail products recommendation. As presented in Table II, the value by using *Absorbing Time* implementation is slightly lower than the *Hitting Time*. This indicates that the more recommended products generated by the implementation of *Absorbing Time* come from the long-tail products. Compared to the result on diversity evaluation that *Hitting Time* probably has the more long-tail products to be recommended, from the result of the long-tail evaluation, it is not valid. Though the long-tail values between both algorithms are slightly different, the *Absorbing Time* performs better. Thus, as the aim of the long tail measurement is to identify whether the recommendation system correctly recommends more long-tail products, it is confirmed for the use of *Absorbing Time*.

VI. Conclusion

This study focus to solve the long-tail problem specifically for B2C e-commerce domain using a tripartite graph representation. Markov random walker is employed to traverse the graph based on *Hitting Time* and *Absorbing Time* algorithm in order to recommend the products for the the customers. The experimental result shows that *Absorbing Time* algorithm yields better accuracy than the *Hitting Time*. The use of this method also slightly generates more long-tail products to be recommended. In terms of diversity, the *Hitting Time* algorithm provides slightly more diverse recommended products. In addition, specialization and generalization on the product category levels as the latent information are observed. The experimental result shows that there is a difference in using generalized vs specialized category levels. *Absorbing Time* perform better in recommendation accuracy combined with the 3-rd level category, and in terms of diversity, the use of this specialized category level for both approaches shows the more diverse recommended products. This experiment shows that to deal with the problem of long-tail in the e-commerce domain, it is possible to make a recommendation by involving the products from the long-tail groups. The diversity score implies that the use of the more specific categories generates the more varied products to be recommended to the users. Through the implementation of the tripartite graph, either *Hitting Time* and *Absorbing Time* approach for graph traversal are considerably to be implemented in B2C companies.

Acknowledgment

This work was supported by Universitas Indonesia through grant "Publikasi Terindeks Internasional (PUTI) Q2: NKB-4060/UN2.RST/HKP.05.00/2020"

References

[1] V. A. Zeithaml, M. J. Bitner, and D. D. Gremler, *Services Marketing Strategy*. American Cancer Society, 2010.

[2] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, "Challenging the long tail recommendation," *Proc. VLDB Endow.*, vol. 5, no. 9, p. 896–907, 2012.

[3] C. Anderson, *The Long Tail Why The Future Of Business Is Selling Less Of More*. New York, NY: Hyperion, 2006.

[4] A. Luke, J. Johnson, and Y.-K. Ng, "Recommending long-tail items using extended tripartite graphs," in *2018 IEEE International Conference on Big Knowledge (ICBK)*, 2018, pp. 123–130.

[5] M. Aprilianti, R. Mahendra, and I. Budi, "Implementation of weighted parallel hybrid recommender systems for e-commerce in indonesia," in *Proceedings of the International Conference on Advanced Computer Science and Information Systems*. Malang - Indonesia: IEEE, 2016, pp. 321–326.

[6] P. H. Aditya, I. Budi, and Q. Munajat, "A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for e-commerce in indonesia: A case study pt x," in *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2016, pp. 303–308.

[7] S. M. Rezaeinia and R. Rahmani, "Recommender system based on customer segmentation (rscs)," *IEEE Transactions on Knowledge and Data Engineering*, vol. 45, no. 6, pp. 946–961, 2016.

[8] F. Rodrigues and B. Ferreira, "Product recommendation based on shared customer's behaviour," *Procedia Computer Science*, vol. 100, pp. 136–146, 2016.

[9] R. Trivonanda, R. Mahendra, I. Budi, and R. A. Hidayat, "Sequential pattern mining for e-commerce recommender system," in *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2020, pp. 393–398.

[10] J. Li, K. Lu, Z. Huang, and H. T. Shen, "On both cold-start and long-tail recommendation with social data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 1, pp. 194–208, 2021.

[11] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, L. Cao, F. Ricci, and P. S. Yu, "Graph learning based recommender systems: A review," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, 2021, pp. 4644–4652.

[12] A. A. Putra, R. Mahendra, I. Budi, and Q. Munajat, "Two-steps graph-based collaborative filtering using user and item similarities: Case study of e-commerce recommender systems," in *2017 International Conference on Data and Software Engineering (ICoDSE)*, 2017, pp. 1–6.

[13] M. Jamali and M. Ester, "Trustwalker: A random walk model for combining trust-based and item-based recommendation," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 397–406.

[14] X. Hu, C. Zhang, M. Wu, and Y. Zeng, "Research on long tail recommendation algorithm," in *Proceedings of the International Conference on Artificial Intelligence Applications and Technologies*, vol. 261. IOP Publishing, oct 2017, pp. 012–019.

[15] Y. Wang, J. Wang, and L. Li, "Enhancing long tail recommendation based on user's experience evolution," in *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, 2018, pp. 25–30.

[16] Y. Wang, J. Wu, Z. Wu, H. Yuan, and X. Zhang, "Popular items or niche items: Flexible recommendation using cosine patterns," in *2014 IEEE International Conference on Data Mining Workshop*, 2014, pp. 205–212.

[17] J. Johnson and Y.-K. Ng, "Using tripartite graphs to make long tail recommendations," in *2017 8th International Conference on Information, Intelligence, Systems Applications (IISA)*, oct 2017, pp. 1–6.

[18] A. Iványi, S. Pirzada, and F. A. Dar, "Tripartite graphs with given degree set," *Acta Universitatis Sapientiae, Informatica*, vol. 7, no. 1, pp. 72–106, 2015. [Online]. Available: https://doi.org/10.1515/ausi-2015-0013

[19] L. László, "Random walks on graphs: A survey, combinatorics, paul erdos is eighty," *Bolyai Soc. Math. Stud.*, vol. 2, 01 1993.

[20] D. Aldous and J. A. Fill, *Reversible Markov Chains and Random Walks on Graphs*, 2002. [Online]. Available: http://www.stat.berkeley.edu/ aldous/RWG/book.html