Evaluating Design Choices in Tripartite Graph-Based Recommender Systems to Improve Long Tail Recommendations

Thomas Crul
Supervisors: Frans Oliehoek, Aleksander Czhechowski, Davide Mambelli, Oussama Azizi
EEMCS, Delft University of Technology, The Netherlands

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Even though the abaility to recommend items in the long tail is one of the main strengths of recommendation systems, modern models still show decreased performance when recommending these niche items. Various bipartite and tripartite graph-based models have been proposed that are specifically tailored to solving this long tail issue. This study aims to investigate the effect of the design of the additional layer introduced by tripartite graph-based recommender systems on their performance. All options available in the MovieLens 1M dataset are evaluated on recall and diversity. Experimental results suggest that tripartite graphs based on latent information describing the users perform better than ones utilising item-based latent information, but both these options hardly outperform the baseline bipartite model. Regardless of the graph used, normalising the transition matrix is found to significantly increase performance. It is hypothesised that larger user-focused additional layers show increased diversity over smaller options when normalised. Issues regarding the reproducibility of previous research are identified and addressed, and the development of unified evaluation metrics is advocated to prevent such problems in the future.

1 Introduction

Recommender systems have become ingrained in a plethora of applications since their introduction. Possibilities for incorporating recommender systems into existing products and services are nearly endless, but specifically the domains of e-commerce and media recommendations have been the main areas of adaptation (Roy & Dutta, 2022). Through decades of research and development recommender systems have been extensively refined and improved, resulting in the wide range of recommender systems present today, as summarized by Bobadilla et al. (2013).

While recommender systems provide numerous benefits like increasing sales in e-commerce settings (Pathak et al., 2010), several issues became apparent through their widespread use. One of these major issues is the fact that where most recommender systems show great performance and accuracy when popular items are concerned, the error rate tends to increase towards the low-ranked items that reside in the long tail of the itemset. This was defined as the *"Long Tail Recommendation Problem"* by Park and Tuzhilin (2008) and will hereafter be referred to as the long tail issue. The notion of the long tail issue was confirmed by Steck (2011) in their empirical study.

Solving the long tail issue is of relevance as recommendations in the long tail are believed to be one of the main driving forces behind the widespread success of recommender systems. Anderson (2008) argues that the increase in sales that was already identified and later confirmed empirically by Pathak et al. (2010) can be largely attributed to recommendations in the long tail. These niche items allow for greater profit margins and can even attract new customers (Luke et al., 2018).

An interesting form of recommender systems that is not covered by Bobadilla et al. (2013) but can prove useful in solving the long tail issue, are graph-based recommender systems. Initially, Yin et al. (2012) pioneered this area by proposing a bipartite graph-based approach towards increasing diversity of recommendations. By applying a random walk across the nodes in the graph to find the items most similar to the preferences of the user, promising results were found when evaluated on the MovieLens dataset (Harper & Konstan, 2015). Subsequently, Johnson and Ng (2017) built on the theoretical foundation constructed by Yin et al. (2012) and extended to a recommender system based on a tripartite graph. Johnson and Ng (2017) reported increased performance after evaluation on the same dataset. This tripartite graph-based solution was later extended by the same authors, which led to a further increase in performance (Luke et al., 2018).

The design choices behind the third group of nodes introduced in the tripartite graph are of crucial importance for the work of Johnson and Ng (2017), as these nodes are the one thing distinguishing their approach from the bipartite graph-based model described by Yin et al. (2012). In their paper, Johnson and Ng (2017) make the design choice to utilise movie genres for this layer of nodes, neglecting the other options present in the data that can be used as a basis to construct this additional layer.

This study aims to contribute to the research towards applying tripartite graph-based recommender systems in solving the long tail issue by pinpointing how design choices governing the construction of the additional layer of nodes influence the performance of the overall system. Specifically, the research question *'What effect does the design of the additional layer of nodes in tripartite-graph based recommender systems have on the overall performance of the system with regards to solving the long tail issue?'* will be answered.

The remainder of this paper is structured as follows. First, Section 2 will provide an overview of the relevant theory and papers crucial to understanding this research. Next, Section 3 describes the approach taken towards answering the research question, and elicits the various options considered for the additional layer of nodes within the constructed tripartite graph used as a basis for recommendations. Subsequently, Section 4 will detail the experimental setup and showcase the results of the performed experiments, which will be discussed in Section 5. Finally, the Responsible Research practices applied during this study will be explained in Section 6, before concluding the paper in Section 7.

2 Background and related work

Various solutions have been proposed that are specifically engineered to solve the long tail issue in recommender systems. This wide range of techniques is extensively covered in the survey paper authored by Qin (2021). Five categories of recommender systems tailored to the long tail issue are identified in the aforementioned paper, but this research limits its scope to focus solely on the *graph based recommendation methods*. This category was pioneered by Yin et al. (2012) through the bipartite graph recommendation model they developed. Subsequently, Johnson and Ng (2017) built on this research by enhancing the bipartite model with a third set of nodes, therefore basing it on a tripartite graph. Finally, Luke et al. (2018) refined the tripartite graph-based model with minor adjustments and improvements across the board, resulting in further increased performance. All three of the aforementioned models will be discussed in sufficient detail in this section.

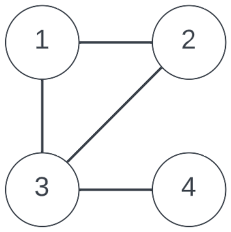

Figure 1: A simple undirected graph

Crucial to understanding both the bipartite and tripartite graph-based approaches are *Markov processes*. This mathematical concept is explained in great detail by Kemeny and Snell (1976), where this section will cherry-pick only the relevant topics needed to understand the research. A Markov process has countless applications in various domains (Pardoux, 2010), but in this specific context, it is used to describe the movement of a *random walker* over a graph. This random walk is performed following probabilities specified in a *transition matrix*. Take, for example, the undirected graph given in Figure 1, which has been adapted from Johnson and Ng (2017). Assuming uniformly distributed transition probabilities, the given graph can be represented by the following transition matrix M (entries are rounded to two decimals for clarity):

$$M = \begin{bmatrix} 0 & 0.5 & 0.33 & 0 \\ \mathbf{0.5} & 0 & 0.33 & 0 \\ 0.5 & 0.5 & 0 & 1 \\ 0 & 0 & 0.33 & 0 \end{bmatrix} \tag{1}$$

The above matrix M contains the probabilities of a random walker reaching node j from node i in a single step (denoted $p_{i,j}$) at entry m_{ij}. The probability of jumping to node 2 from node 1, for example, is highlighted in bold in M. Worth noting is the fact that the entries in each column of a transition matrix must sum up to 1, seeing as it encompasses the probabilities of all options of the random walker at a given node.

Transition matrices can be used to find the distribution of the position of a random walker after an arbitrary amount of steps t, by raising the transition matrix to the power t. For example, the distribution of the position of a random walker traversing the graph shown in Figure 1 for two steps is as follows (again, entries are rounded to two decimals for clarity):

$$M^2 = \begin{bmatrix} 0.42 & 0.17 & 0.17 & 0.33 \\ 0.17 & 0.42 & 0.17 & 0.33 \\ 0.25 & 0.25 & 0.66 & 0 \\ 0.17 & 0.17 & 0 & 0.33 \end{bmatrix} \tag{2}$$

When starting at node 1, there is only one way to end up in node 3 after two steps: jump to 2 with a 50% probability, then jump to 3 with a 50% probability. Hence for M^2, $m_{13} = 0.25$.

For higher values of t, the resulting matrix will converge to the so-called *stationary distribution*. When this distribution has been reached, additional steps of the random walker will not change the distribution of its position. For the graph shown in Figure 1, the random walk converges to its stationary distribution after 38 steps:

$$M^{38} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.375 & 0.375 & 0.375 & 0.375 \\ 0.125 & 0.125 & 0.125 & 0.125 \end{bmatrix} \tag{3}$$

Interesting here is the fact that for any starting node in the graph, the random walker is most likely to end up in node 3 given enough time. In general, uniform random walks favour nodes with the highest *degree*, which is the number of edges a particular node has. This is of crucial importance in the context of graph-based recommender systems, as nodes with the lowest degree are rated the least and therefore reside in the long tail.

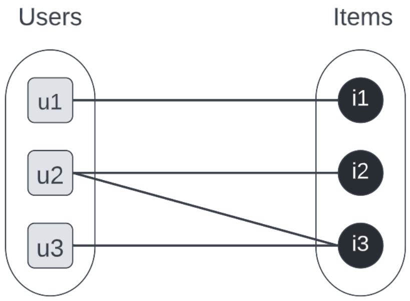

Figure 2: A bipartite user-item graph

The *bipartite graph-based model* incorporates two sets of nodes into its undirected graph: set U containing the users, and set I containing the items. A visual representation of this bipartite user-item graph is shown in Figure 2. A user u is connected to an item i if and only if the user rated the item, and the weight of this edge is proportional to the rating given to the item by the user. To find items to recommend for a query user q, Yin et al. (2012) propose three concrete algorithms; Hitting Time, Absorbing Time, and Absorbing Cost. All three algorithms aim to find items with few ratings relevant to the query user, therefore mitigating the long tail issue.

The *Hitting Time (HT)* between nodes j and q, denoted as $H(q|j)$ is defined as "the expected number of steps that a random walker starting from node j ($j \neq q$) will take to reach node q" (Yin et al., 2012). Recommendations are generated by computing $H(q|j)$ for all $j \in I$ not rated by q and taking the item nodes with the lowest HT. This intuitively returns the set of items 'closest' to query user q in the bipartite graph. The initial probability p_{ui} to jump from user u to item i and vice versa is given by Formula 4, where $r(u, i)$ is the rating user u gave to item i.

$$p_{ui} = \frac{r(u,i)}{\sum\limits_{k \in I} r(u,k)} \tag{4}$$

The transition matrix containing these probabilities is then used to calculate the stationary probability of the random walk. Subsequently, the HT of two nodes is calculated following Formula 5, where π_j equals the stationary probability of node j, and $p_{q,j}$ is the stationary probability of a random walker reaching node j when it starts in node q.

$$H(q|j) = \frac{\pi_j}{p_{q,j}\pi_q} \tag{5}$$

Therefore, the HT algorithm prioritises items with a low stationary probability π_j and a high transition probability to the query user $p_{q,j}$. In other words, this algorithm recommends items that both have few ratings, and are relevant to the query user. Relevancy between a user and item increases as there are more paths connecting them, these paths become shorter, and the weights (and thus the ratings) along the edges constituting the paths increase. These properties effectively combine to elegantly solve the long tail issue.

The *Absorbing Time (AT)* algorithm generalises the HT algorithm by introducing a set S consisting of *absorbing nodes*. When generating recommendations for a query user q, these absorbing nodes are chosen to be all items previously rated by q. Yin et al. (2012) define AT as follows: "Given a set of absorbing nodes S in a graph G, the absorbing time denoted as $AT(S|i)$ is the expected number of steps before a random walker, starting from node i, is absorbed by S.". In other words, instead of calculating the proximity of items to the query user like in the HT algorithm, the AT algorithm calculates the distance to the set of items rated by the query user. AT is a generalisation of HT, as when $S = \{q\}$, the two algorithms are equivalent.

The *Absorbing Cost (AC)* algorithm further extends the AT algorithm by incorporating additional information into the model beyond the ratings given by users. In essence, the AC algorithm introduces weights to user ratings, valuing the ratings of users with a more specific taste higher than that of users with more broad preferences in items. The specificity of user tastes is measured by the *entropy* of their preferences. The first form of the AC algorithm, dubbed *AC1*, calculates the entropy based on the items they rated, while the second form of the AC algorithm, aptly named *AC2*, analyses the distribution of the user over a set of *topics*.

Yin et al. (2012) reported that out of the algorithms they developed, the AC2 algorithm performs best in terms of *accuracy*, while the AC1 algorithm performs best when *diversity* is concerned.

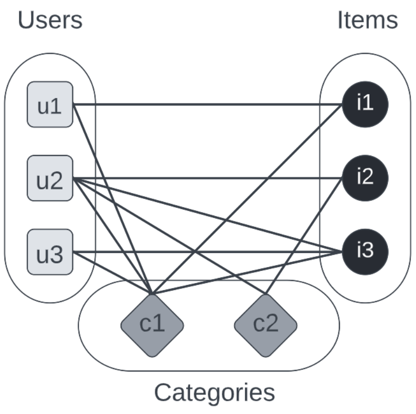

Figure 3: A tripartite user-item-category graph

The *tripartite graph-based model* initially developed by Johnson and Ng (2017) incorporates additional latent information into the bipartite graph-based model through a third set of nodes C, as shown in Figure 3. The aim of this additional *intermediate layer* is to fill in the gaps when available user-item data is too sparse, and provide additional short paths when traversing the graph, resulting in improved recommendations. Like the bipartite graph-based model previously discussed, this model was evaluated on the MovieLens dataset. Latent information regarding the genres of the movies in this dataset was chosen to form the basis for the newly introduced layer of nodes. Here, the *full genres* were taken; the entire set of genres a movie falls under is considered, instead of splitting them into *basic genres*. This difference is further illustrated in Figure 4[1]. As a result of this design choice, an item can only be connected to a single node in the additional layer.

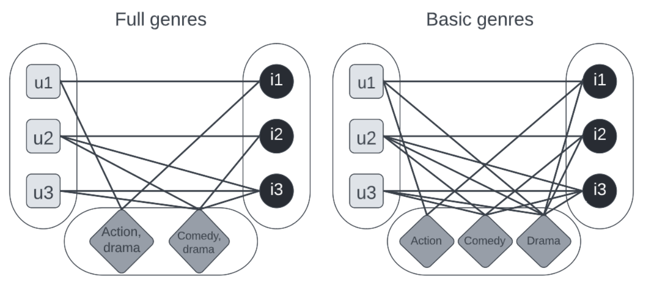

Figure 4: An example illustrating the difference between full and basic genres

Like for the bipartite graph-based model discussed previously, the weights given to the edges present in the graph are of crucial importance for the performance of the overall recommender system. The edges connecting users and items are given the same weight in both the bipartite and tripartite model, as described in Formula 4. User u is connected to genre g following the formula described in Formula 6 , where $ar(u, g)$ is the average of all the ratings user u gave to the items in genre g.

[1]This figure contradicts the explanation of Qin (2021), which misplaces the intermediate layer

$$w_{u,g} = \frac{ar(u,g)}{\sum\limits_{c \in C} ar(u,c)} \tag{6}$$

These edge weights ensure that the genres rated the highest by a given user are prioritised over others. Likewise, Johnson and Ng (2017) chose to connect item i to its genre g with weight $w_{i,g}$ shown in Formula 7, where $ar(j)$ is the average of all ratings given to item j.

$$w_{i,g} = \frac{ar(i)}{\sum\limits_{j \in g} ar(j)} \tag{7}$$

In other words, when a random walker is at a given genre, it will favour stepping to items with the highest average rating. Here, the crucial assumption is that items with higher ratings on average are better to recommend than items scoring lower.

To generate recommendations in the tripartite graph-based model, the HT algorithm is simulated through a Markov process. Again, the random walk is started from a query user, but instead of converging to the stationary distribution, only a limited number of steps are taken. This circumvents the issue of favouring highly connecting nodes, which would lead to decreased recommendations in the long tail. To this end, Johnson and Ng (2017) defined the *T3*, *T5* and *T7* algorithms, where the random walker performs three, five and seven steps, respectively. All three of these algorithms pick the items not rated yet by the query user with the maximal probability of arrival by the random walker as recommendations. Out of these three options, T5 performs best, and both T3 and T5 edge out the bipartite AC algorithms in terms of accuracy, but both algorithms perform worse when diversity is considered.

In their paper, Luke et al. (2018) describe three improvements to the tripartite graph-based model. First, like illustrated in Figure 4, the *full genres* were split up in favour of *basic genres*. Second, the weights of the edges connecting these basic genre nodes were updated to utilise the *Bayesian average* shown in Formula 8.

$$w_{u,g} = \frac{av(u) * ar(u) + v(u,g) * ar(u,g)}{av(u) + v(u,g)} \tag{8}$$

In Formula 8, $av(u)$ is the average number of items user u rated per genre, $ar(u)$ is the average rating user u gave per genre, $v(u,g)$ is the amount of items user u rated in genre g, and $ar(u,g)$ was previously defined in Formula 6. Finally, the edge weights connecting basic genre nodes to item nodes adjusted to follow Formula 9, where $ar(i)$ is the average rating of item i and G_i is the set of basic genres applicable to i.

$$w_{i,g} = \begin{cases} \frac{ar(i)}{|G_i|} & \text{if } g \in G_i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Through these improvements the T3 algorithm outperforms all other previously discussed options both in terms of accuracy and diversity.

3 Methodology

This research aims to investigate the influence of the design of the additional layer of nodes introduced in the tripartite graph-based recommender system model on its performance. Previously, Johnson and Ng (2017) and Luke et al. (2018) reported on the performance of *full* and *basic genres*, but to the researcher's best knowledge no further research has been completed towards this topic.

The additional layer introduced in the tripartite graph-based model can be based on latent information describing either the items or the users. For both of these approaches, the same baseline bipartite graph containing solely the user and items nodes is utilised. This is the bipartite graph described by Yin et al. (2012), where edges connecting users to items are connected following Formula 4. The sole exception to this is the tripartite graph based on *full genres* described by Luke et al. (2018), as this uses different weights between users and items. Here, users are simply connected to items by the rating they have given it, with zero being the default.

For the graphs based on categorical latent information about the items, the edges connecting the additional layer will be weighted following the method described by Johnson and Ng (2017). These edge weights described in Formula 6 and Formula 7 were chosen over the improvements implemented by Luke et al. (2018), as this latter approach was developed for a many to many relationship between items and categories, which normally is not applicable.

A different approach has to be taken for the tripartite graphs focused on the latent information about users, as these have a substantially different structure. Here, user nodes are connected to additional nodes with weight $w_{u,c}$ shown in Formula 10, where U_c is the set of users belonging to additional category c. This method ensures that all outgoing transitions from categories to users sum op to one, which has the added side effect of discounting large categories. In other words, the collective preferences of large groups that are more broad and general are valued less than smaller communities the users are part of.

$$w_{u,c} = \begin{cases} \frac{1}{|U_c|} & \text{if } u \in U_c \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Similar to the weights shown in Formula 7, additional nodes are connected to item nodes following Formula 11, where $ar(c,i)$ is the average of all ratings of item i from users belonging to additional category c. These edge weights ensure that items preferred by the collective making up the category likewise are prioritised by the random walker when generating recommendations.

$$w_{c,i} = \frac{ar(c,i)}{\sum\limits_{j \in I} ar(c,j)} \tag{11}$$

To generate recommendations from the various graph options, three different algorithms will be tested. The *T3*, *T5*, and *T7* algorithms all employ a random walker that respectively takes three, five and seven steps when generating recommendations for a query user. These algorithms were first developed by Johnson and Ng (2017), and based on their results the T3 algorithms is expected to outperform both other options.

Additionally, a potential point of improvement to the existing models that will be tested is the normalisation of the transition matrices. These matrices form the basis for the Markov process employed on the constructed graphs and therefore largely influence the performance of the overall model. As explained previously in Section 2, the values in the columns of the transition matrix should sum up to one, as this exhaustively covers all the traversal options a random walker has at a given node. The issue with the methods described by Yin et al. (2012), Johnson and Ng (2017) and Luke et al. (2018) is the fact that the adjacency matrices resulting from their weighting formulas violate this property. To investigate the impact of this design flaw on the performance of the models additional experiments will be run using normalised adjacency matrices as a basis. This normalisation is performed by dividing all values in a column by the sum of its values.

4 Experimental Setup and Results

Like the bipartite and tripartite graph-based approaches described previously in Section 2, this research utilises the MovieLens 1M dataset (Harper & Konstan, 2015). This dataset consists of 1,000,209 ratings on a scale of 1 through 5 on 3,952 movies by 6,040 users. Additionally, demographic data concerning *gender*, *age*, *occupation*, and *zip code* is included for every user. Here, the distinction is made between 2 genders, age is aggregated in 7 ranges, the occupation is covered by 21 labels, and 3,439 distinct zip codes are included. Latent information about the movies includes *release year* - covering a span of 81 years - and *genres* - consisting of 301 unique combinations of 18 genres.

To effectively answer the research question, all options for the additional layer in the tripartite graph-based model will be evaluated on their performance. As a benchmark, the bipartite model foregoing any additional layer described by Yin et al. (2012) and the additional layers encompassing *full* and *basic* genres implemented previously by respectively Johnson and Ng (2017) and Luke et al. (2018) are included in evaluation. Additionally, the following five options are considered for the additional layer:

- The *gender* of the user, counting 2 options.
- The *age* of the user, counting 7 options.
- The *occupation* of the user, counting 21 options.
- The *zip code* of the user, counting 3,439 options.
- The *release year* of the movie, counting 81 options.

Thus, three options incorporating item-based latent information and four options based on user-focused latent information are evaluated in addition to the basic bipartite model.

4.1 Evaluation Metrics

To evaluate the performance of the various options for the additional layer in the tripartite graph-based model, the same metrics will be applied as by Yin et al. (2012), Johnson and Ng (2017), and Luke et al. (2018).

First, *Recall@N* aims to effectively indicate the accuracy of the model in recommending items that are actually relevant to the user. In order to evaluate the tripartite models on *Recall@N*, the following procedure is followed:

1. Create test set T by removing 4,000 five star ratings from the training set at random.
2. For each test item $t \in T$ randomly select 1,000 items not rated by the user who created rating t.
3. Use the training set to order the 1,001 selected items in order of relevance to the query user.
4. Calculate *Hit@1* through *Hit@50*, as defined in Formula 13, where t is the test item defined in step 2 and $rank(t)$ is the rank of item t in the ordered list constructed in step 3.

$$Hit@N(t) = \begin{cases} 1 & \text{if } rank(t) \leq N \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

5. Aggregate the results of all items in the test set into *Recall@1* through *Recall@50*, as defined in Formula 13.

$$Recall@N = \frac{\sum\limits_{t \in T} Hit@N(t)}{|T|} \tag{13}$$

By removing the five-star ratings from the data provided, the model's ability to effectively learn the preferences of the users and generate recommendations accordingly is tested.

Additionally, the models are evaluated on the *Diversity* metric to get a grasp of their effectiveness in solving the long tail issue. This metric captures the intuition that a recommender system that recommends the same popular items to all users performs worse in promoting items in the long tail than a system that shows many unique items to its users. Diversity is calculated as defined below in Formula 14, where U is the set of users, R_u is the set of items recommended to user u, and I is the set of items.

$$Diversity = \frac{\bigcup\limits_{u \in U} R_u}{|I|} \tag{14}$$

In the experiments conducted during this research, U was chosen to consist of 200 random test users and R_u to equal the top 10 recommendations for user u. Therefore, the maximum amount of distinct recommendations possible is 2000, which is the value of $|I|$.

4.2 Results

This subsection shows the results most relevant for answering the research question posed in this paper. Details on the experiments, raw data and additional results not detailed here can be found in the repository containing all code used over the course of this research[2].

[2]The code repository can be accessed via: https://gitlab.com/ThomasCrul/evaluating-tripartite-recommenders

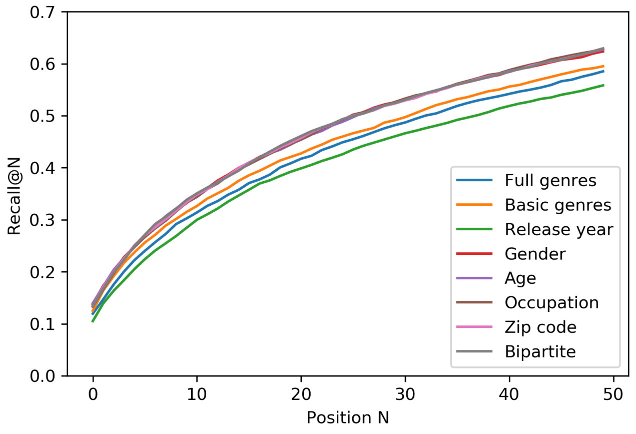

Figure 5: The performance of the T3 algorithm on the graph options

Figure 5 shows the performance of the graph options, as measured by the *Recall@N* metric. Here, only the performance of the T3 algorithm is shown for each option, as this algorithm consistently outperformed the T5 and T7 algorithm. All three of the options constructed with latent information describing the items showed marginally decreased performance, while the remaining five options demonstrated nearly identical performance.

Due to the stochasticity introduced by sampling 200 users at random in its calculation, the value of the *Diversity* metric fluctuates between experiments. To counteract this, the average of 100 Diversity calculations was taken and the variance of these samples was used to construct 95% confidence intervals, which are shown by the black bars in Figure 6 and Figure 8. Randomness is, in fact, introduced in the calculation of the *Recall@N* metric as well by arbitrarily selecting 1,000 items not rated by the query user. This is however averaged over 4,000 test cases, and as a result this metric hardly fluctuates between experiments. When averaged over just three calculations, the 95% confidence intervals already are barely visible in the resulting graph, therefore they are not included. The lines shown in Figure 6 and Figure 8 are the results of single experiments.

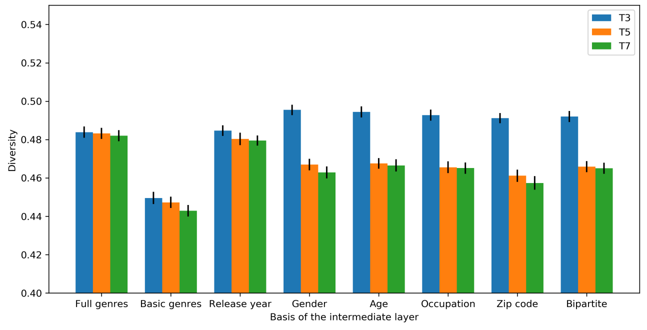

Figure 6: The diversity of the graph options and algorithms

Figure 6 shows the diversity of all graph options, achieved by the three algorithms performed on them. Interesting again is the split in performance between the item-based latent information options and the rest. All three of these options show decreased diversity, and rather constant scores between the algorithms, where the other options show marginally increased diversity for the T3 algorithm, which rapidly decreases for both T5 and T7.

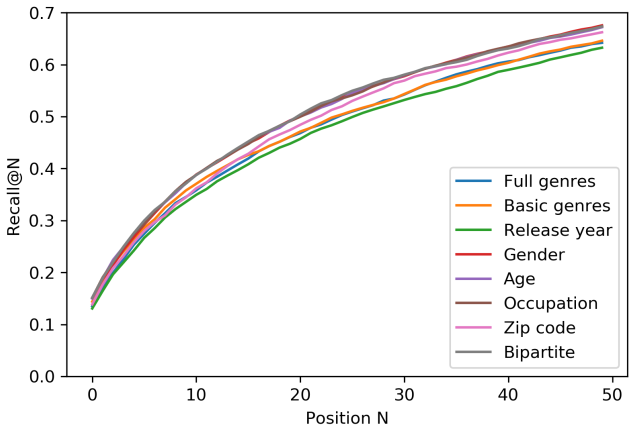

Figure 7: The performance of the T3 algorithm on the normalised graph options

Figure 7 and Figure 8 detail the performance and diversity of the *normalised* graph options. Remarkable in both figures is the fact that the simple act of normalising the transition probabilities shows significant increases in both evaluation metrics for all graph options. For recall, Figure 7 shows that the options formed with item-based latent information again perform worse than the other options, with the *zip code* graph now lagging just behind the rest of the pack in terms of performance, albeit with only a few percentage points. As for diversity, Figure 8 shows that all three of the options utilising latent information about the items show decreases in diversity between the algorithms in line with the rest of the graph options. Most remarkable however is the jump in diversity of the *zip code* graph, which now edges out all other options.

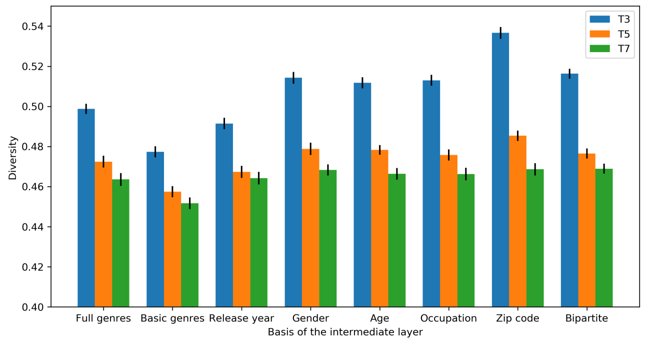

Figure 8: The diversity of the normalised graph options and algorithms

5 Discussion

The results shown previously in Section 4 indicate that the design of the additional layer in the tripartite graph-based does, in fact, impact the model's performance. It seems that within the design choices for this extra added layer, the decision to base it either on latent information describing the items or on information about the users is the most influential. All four user-based graph options outperformed the item-based graph options both in terms of recall and diversity, albeit with a slim margin. No differences in performance were found between the four user-based graph options without applying normalisation, even though the amount of nodes in their additional layer ranges from just two to almost four thousand nodes. This seems to indicate that the number of nodes in the additional layer has little effect on the overall performance of the model when normalisation is not applied.

Most striking however, is the fact that the bipartite graph-based model shows comparable performance to the various tripartite graph-based options. This more simple approach achieved similar results to the graph options based on user-centered latent information, therefore outperforming the item-based graph options. This raises the question whether it is necessary to introduce the intermediate layer into the original bipartite model in the first place. Within the context and limitations of this research, few advantages of the addition of the intermediate layer were found, while increasing computational overhead of the model. In light of this tradeoff, the results of this research suggest that it is inadvisable to add unnecessary complexity to the model through the intermediate layer.

The most concrete takeaway of the experiments is the positive impact of the normalisation of the transition matrices. This computationally inexpensive preprocessing step increased the performance of every single graph option in both metrics. By adhering to the requirements stipulated for transition matrices in Markov processes, individual nodes are prevented from being assigned inflated transition probabilities. These nodes would otherwise dominate the random walk performed on the graph, and therefore skew the recommendation process in their favour.

Out of all the options for the additional layer, the graph based on the *zip codes* of the user saw the greatest difference in performance through the application of normalisation. This stands to reason, as this graph option has the largest amount of nodes in its additional layer. As the transition probabilities from these nodes to the users and items are normalised with respect to themselves, each node adds excess transition probabilities to both sets that sum up to one. This inflation of the transition probabilities of the user and item nodes, therefore, is proportional to the number of nodes present in the additional layer that is introduced. The higher the inflation of the transition probabilities, the more effect normalisation of the transition matrix has.

After applying normalisation, the zip code graph shows a remarkable jump in performance in the diversity metric. Intuitively, recommending movies based on the area of residence of users does not immediately strike as a logical choice, as the taste of a user can vary wildly from the taste of its neighbour. This intuition is reflected in the fact that utilising the zip code over other user-centered latent information options comes at a minor expense of accuracy, as shown in Figure 7. When considering diversity however, a significant jump in performance can be seen over the other graph options. A possible explanation for this discrepancy is the number of nodes that make up the zip code layer, as it is more than 163 times as large as the occupation graph, which contains the second largest user-focused additional layer. The performance of the normalised graph options suggest that additional layers of increased size significantly improve diversity, while marginally decreasing their accuracy. Further research extending to additional datasets is needed to confirm this hypothesis.

Through the experiments performed in this research, previous results could not always be replicated. Despite a thorough analysis of the research of Luke et al. (2018), the reported performance improvement of the *basic genres* graph over the *full genres* graph did not show in experiments. This is most likely attributable to differences in concrete implementations of the models between this research and that of Luke et al. (2018), as important details were left out on multiple occasions. An example of this is the fact that the basic genre graph utilises different weights between users and items than the full genres graph, which had to be rediscovered during this research as this is never mentioned in the paper describing it. Without this change in user-item weights, the model showed abysmal performance.

This issue of reproducibility was not just relevant to the implementations of the models, but also to the realisation of the evaluation metrics. For the *Recall@N* metric, Yin et al. (2012) write that they select 4,000 *long tail ratings* at random as the test set, without specifying what is defined as such. Next, Johnson and Ng (2017) merely write that the dataset is split into a training set and a test set, without providing any further details whatsoever. Finally, Luke et al. (2018) randomly sample an unknown amount of users to make up the test set, instead of ratings. Likewise, the implementations of the *Diversity* metric are inconsistent between the papers. Where Yin et al. (2012) sampled 2,000 users to calculate the diversity score, Luke et al. (2018) take just 200. This last paper also mentions that in calculating their diversity scores, Johnson and Ng (2017) mistakenly recommended items users had previously rated. Considering these ambiguities and differences in the implementations of the evaluation metrics between the referenced papers, it becomes impossible to effectively compare their results, as there is no common benchmark. This problem of irreproducible performance metrics was previously identified and substantiated by Qin (2021). Their survey paper similarly concludes that recommender systems tailored to solving the long tail issue are lacking in unified evaluation metrics.

Finally, a reservation about the scalability of the graph-based recommender system models past static datasets has to be made. Due to the definitions of the edge weights in both the bipartite and tripartite graph-based models, all ratings need to be aggregated in order to compute them. As a result all edge weights in the graph need to be recomputed whenever new ratings are introduced into the system. Additionally, the addition of new users and items is problematic as this changes the dimensions of the adjacency matrix representing the graph. As a sidenote, newly introduced items would never be recommended to users, as long as they do not receive a rating from at least one user. Recomputing all edge weights and changing the dimensions of the adjacency matrix both pose significant challenges to overcome before the graph-based recommender models can be applied to real-world systems, as these require high adaptability. Finding solutions to these challenges is a fruitful area for further research.

6 Responsible Research

As previously highlighted in Section 5, various problems were encountered while trying to implement models previously described in the literature. Both the models themselves and the performance criteria they were evaluated by were described in an insufficient level of detail to replicate the reported results. This issue is a known and prevalent problem within the scientific community, as explained by Baker (2016). Raghupathi et al. (2022) researched this reproducibility crisis specifically in the context of computing research and formulated 25 criteria for the reproduction of scientific research in doing so. To ensure that the research described in this paper can easily be reproduced by fellow researchers, all of the 25 aforementioned factors have been included in this paper.

While this research promotes the application of recommendation systems, one should be aware of the fact that they are not free of negative ethical ramifications. Milano et al. (2020) distilled six main ethical issues for recommendation systems based on existing literature; concerns regarding inappropriate content, privacy, autonomy, opacity, fairness and social effects. Similarly, Polonioli (2021) identified comparable problems through their research of the ethical consequences of scientific recommender systems. The models shown in this research are not free from these ethical complications. Therefore, these issues should be carefully considered when developing a recommendation system based on the results of this research that will interact with real users.

In contrast, solving the long tail issue can potentially mitigate one of the ethical issues recommendation systems are currently facing. Modern recommendation systems suffer from popularity bias, where the recommendations favour items that are already popular, reinforcing their popularity (Bellogín et al., 2017). This results in a winner-takes-all scenario, where a small number of items gets recommended to the majority of users (Milano et al., 2020; Polonioli, 2021). Improving the performance of recommendation systems in the long tail will increase the diversity of recommendations and mitigate this undesirable winner-takes-all scenario. However, even after solving the long tail issue ethical issues regarding the application of recommendation systems remain.

7 Conclusions and Future Work

This paper set out to investigate the influence of the design of the additional layer in the tripartite graph-based model for recommendation systems described by Johnson and Ng (2017) on its performance. Based on experimental results limited to the MovieLens 1M dataset (Harper & Konstan, 2015), it can be concluded that basing the additional layer on latent information describing the user yields the best results, both in terms of recall and diversity. Simultaneously, it was found that the simple baseline bipartite graph-based model created by Yin et al. (2012) shows comparable results to the optimal tripartite graph options, foregoing the need to introduce the third layer into the model to begin with. Regardless of the makeup of the graph, normalising the transition probabilities to conform to stipulations formulated for transition matrices in Markov processes yielded increased performance for both applied performance metrics.

Four main directions for further research were identified. First, the tripartite-graph based model can be extended to utilise a multipartite model by combining various graph options. Next, this research advocates for the development of unified metrics to evaluate recommender systems focused on the long tail, following from the issues regarding the reproducibility of previous research that were encountered. Third, the experiments can be extended to additional datasets to test the hypothesis that larger user-focused additional layers show increased diversity over smaller options when normalised. Finally, challenges hindering the real-world applicability of the graph-based model remain unsolved.

References

Anderson, C. (2008). *The Long Tail: Why the Future of Business is Selling Less of More*. Hyperion.

Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, *533*(7604), 452–454. https://doi.org/10.1038/533452a

Bellogín, A., Castells, P., & Cantador, I. (2017). Statistical biases in Information Retrieval metrics for recommender systems. *Information Retrieval Journal*, *20*, 1–29. https://doi.org/10.1007/s10791-017-9312-z

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132. https://doi.org/https://doi.org/10.1016/j.knosys.2013.03.012

Harper, F. M., & Konstan, J. A. (2015). The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*, *5*(4). https://doi.org/10.1145/2827872

Johnson, J., & Ng, Y.-K. (2017). Enhancing Long Tail Item Recommendations Using Tripartite Graphs and Markov Process. *Proceedings of the International Conference on Web Intelligence*, 761–768. https://doi.org/10.1145/3106426.3106439

Kemeny, J. G., & Snell, J. L. (1976). *Finite Markov Chains*. Springer-Verlag.

Luke, A., Johnson, J., & Ng, Y.-K. (2018). Recommending Long-Tail Items Using Extended Tripartite Graphs. *2018 IEEE International Conference on Big Knowledge (ICBK)*, 123–130. https://doi.org/10.1109/ICBK.2018.00024

Milano, S., Taddeo, M., & Floridi, L. (2020). Recommender systems and their ethical challenges. *AI & SOCIETY*, *35*(4), 957–967. https://doi.org/10.1007/s00146-020-00950-y

Pardoux, É. (2010). *Markov Processes and Applications: Algorithms, Networks, Genome and Finance*. https://doi.org/10.1002/9780470721872

Park, Y.-J., & Tuzhilin, A. (2008). The Long Tail of Recommender Systems and How to Leverage It. *Proceedings of the 2008 ACM Conference on Recommender Systems*, 11–18. https://doi.org/10.1145/1454008.1454012

Pathak, B., Garfinkel, R., Gopal, R., Venkatesan, R., & Yin, F. (2010). Empirical Analysis of the Impact of Recommender Systems on Sales. *J. of Management Information Systems*, *27*, 159–188. https://doi.org/10.2307/29780174

Polonioli, A. (2021). The ethics of scientific recommender systems. *Scientometrics*, *126*(2), 1841–1848. https://doi.org/10.1007/s11192-020-03766-1

Qin, J. (2021). A Survey of Long-Tail Item Recommendation Methods (D. Hong, Ed.). *Wireless Communications and Mobile Computing*, *2021*, 7536316. https://doi.org/10.1155/2021/7536316

Raghupathi, W., Raghupathi, V., & Ren, J. (2022). Reproducibility in Computing Research: An Empirical Study. *IEEE Access*, *10*, 29207–29223. https://doi.org/10.1109/ACCESS.2022.3158675

Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1), 59. https://doi.org/10.1186/s40537-022-00592-5

Steck, H. (2011). Item Popularity and Recommendation Accuracy. *Proceedings of the Fifth ACM Conference on Recommender Systems*, 125–132. https://doi.org/10.1145/2043932.2043957

Yin, H., Cui, B., Li, J., Yao, J., & Chen, C. (2012). Challenging the Long Tail Recommendation. *Proc. VLDB Endow.*, *5*(9), 896–907. https://doi.org/10.14778/2311906.2311916