Model Performance Estimation Using the Dual Augmented Two-towers

Michael Cornelisse
s1059020
Radboud University
Nijmegen, Netherlands
michael.cornelisse@ru.nl

Felix Jaspersen
s1041254
Radboud University
Nijmegen, Netherlands
felix.jaspersen@ru.nl

Yme Kingma
s1020242
Radboud University
Nijmegen, Netherlands
yme.kingma@ru.nl

Abstract

Each year, many articles are published about information retrieval with many proposals for improvements in different tasks. Before rushing to implement the latest state-of-the-art system you may want to evaluate if the model is even an improvement, or if the model works at all. To this end, we assess the performance of the Dual Augmented Two-tower model proposed by Yu *et al.* [6] using bootstrapping on an Amazon books dataset. Our results show similar results for the hitrates at 50, and 100, but not for the MRR. However, this can be explained by the fact that our test set has fewer items, so items that perform very badly cannot deteriorate the metric as much. Furthermore, our experiment supports the claim by Yu *et al.* that their Category Alignment Loss improves the model's performance.

CCS Concepts

• **General and reference** → **Verification**; **Estimation**; • **Information systems** → *Recommender systems*; Information retrieval.

Keywords

Reproducing Paper, Estimation methods, Recommender Systems, Information Retrieval, Neural Networks

1 Introduction

With the ever-increasing amount of knowledge and content available on the web, in-house databases of companies, and institutional repositories, narrowing down the range of results for a user has become critically important. This process is typically described in two phases. The first phase, known as the retrieval phase, gathers relevant results from a large corpus of items, documents, or other entities, depending on the application. The second phase involves ranking these results in a sensible order based on relevance, ensuring that the user is presented with the most useful outcomes early on. Naturally, the effectiveness of the ranking phase depends heavily on the quality of results retrieved in the first phase.

There are two prominent contexts to consider. That of a search engine where a user actively searches with a query for specific items or documents and that of a recommender system. The latter tries to anticipate relevant items without a query but given data about the user. The technical solutions to this problem are referred to as recommender systems. The simplest approach is memory-based collaborative filtering, where a user-item matrix is created for all possible combinations, and similar pairs are matched. However, this method is severely limited in terms of scalability and suffers from sparsity in the data. As a result, more sophisticated recommender systems have been developed that rely on latent representations instead of direct pair matching.

Two examples of such advanced models are Matrix Factorization and the Two-Tower System. The latter model is a particularly interesting architecture because it blends collaborative filtering with a content-based approach. It employs two neural networks to learn latent representations — one for users and another for the items themselves. These latent spaces are then aligned such that a vector in the user space coincides with a corresponding item vector when the pair forms a good match for a recommendation.

In 2021, Yu *et al.* proposed an improved version of the two-tower model in their paper A Dual Augmented Two-tower Model for Online Large-scale Recommendation [6]. They introduced two key innovations aimed at addressing limitations in the original architecture.

The first innovation is the **Category Alignment Loss (CAL)**, which addresses the imbalance across item categories. This mechanism transfers knowledge learned from categories with a large amount of data to those with less data, thereby aiming to improve the model's performance across all categories. The authors describe it as "transferring knowledge from the category with the largest amount of data to other categories".

The second improvement is the **Adaptive-Mimic Mechanism (AMM)**, which enhances the original item and user embeddings by introducing an additional feature that captures "historical positive interactions". For a given user, this mechanism generates a feature that represents all the items the user has interacted with, not just their IDs but their latent representations from the item tower. Similarly, an augmented vector is created on the item side. On the user side, this feature can be thought of like a buyers' history.

Having said this, this paper answers the following research question. *Can we reproduce Yu et al.'s offline evaluation results on the Amazon Books dataset using a computationally efficient approach like bootstrapping?*. Reproducing scientific papers plays a crucial role in ensuring that research results are reliable and verifiable. It can also uncover potential mistakes, gaps, or optimization opportunities in the original work. Additionally, reproducing private research provides an opportunity to share previously unavailable methodology, such as code and implementation details.

We explore whether a lightweight bootstrapping approach can be used to reproduce the results from Yu *et al.* on the Amazon Book dataset [4]. Such a strategy is beneficial as applications like recommender systems typically require large-scale data. Simulating these systems with smaller datasets can provide a more practical and resource-efficient way to verify complex models like Yu *et al.*'s.

Details on the architectures, methodologies, and findings will follow in the subsequent sections.

2 Related work

This project builds predominantly on the work of Yu *et al.*, which extended the Two-Tower model with new mechanisms to address category imbalance and improve tower interactions. Additionally, we would like to acknowledge two types of related research: the paper that introduced the original Two-Tower model and more recent developments that leverage and extend this architecture.

2.1 Introduction of the Two-Tower Model

The foundational concept of the Two-Tower model can be traced back to the work of Huang *et al.* in 2013 [3]. Although their paper did not explicitly introduce the term "Two-Tower model", their approach laid the groundwork for it. Their work addressed a classic information retrieval (IR) problem: matching queries with documents.

At the time, latent semantic models existed to map words to a latent space, capturing their semantic meaning to improve query-document matching. Huang *et al.* improved upon these models by mapping both queries and documents into a shared low-dimensional space. Simplifying the actual procedure, they trained two separate linear neural networks to generate latent spaces for queries and documents and updated the networks to minimize the angle (cosine similarity) between correct query-document pairs in this latent space.

An additional innovation in their work was the introduction of a precursor to modern embedding layers, which they called "word hashing". This method used a 30,621-dimensional n-gram representation to map a dictionary of approximately 500,000 entries. Their architecture significantly improved search relevance results compared to traditional approaches like term matching, paving the way for future Two-Tower models.

2.2 Dual Augmented Two-Tower Model

As mentioned before Yu and colleagues, extended the two-tower approach with an AMM to enhance tower interaction, as well as a CAL that aims to unify relationships of the latent variables of the items embedding.

In this project, we focused on reproducing the offline results of Yu *et al.* on the Amazon Books dataset, which consists of 6,706,125 interactions. However, Yu *et al.*'s paper went further, testing their model on the much larger Meituan dataset, which contains over a billion interactions and demonstrated its effectiveness in an online use case involving 60 million users per day over a week.

2.3 Recent Developments

The Two-Tower architecture has gained widespread popularity due to its intuitive structure and good results. Therefore, it is no surprise that one can find recent additions to this idea.

One notable addition was introduced by Cui *et al.* [1] earlier this year, incorporating **multimodality** into the Two-Tower model. Their approach combines textual and visual features by using a Bidirectional Long Short-Term Memory (BiLSTM) network to extract text features and an External Attention Transformer to extract image features. These feature vectors are concatenated with the standard item embeddings, enriching their representational capacity.

Table 1: Statistics of the Datasets

Dataset	#users	#items	#interactions	#categories
Yu *et al.*	695,513	243,166	6,706,125	11
Raw	603,668	367,982	8,898,041	94
Final	347,084	161,624	4,997,292	11

3 Method

In order to reproduce the results from the paper, we followed their description as closely as possible. In this section we will highlight any unclarities and questionable practices in the paper and how we interpreted these to approximate their approach as best as we can.

3.1 Dataset

Yu *et al.* used two datasets to evaluate their model: a large dataset from Meituan, a Chinese shopping platform consisting of daily logs of their online systems, and the book review data set from Amazon [4]. Yu *et al.* stated that they "only kept the items that have been reviewed for at least 5 times and users with who have reviewed at least 5 items". To stay consistent with their approach we specifically used the 5-core version, a dense subset containing only users and items with at least five reviews each.

The dataset consists of two parts: the book reviews (reviewerID, rating, text, etc.) and metadata (title, description, category information, price, etc.). The exact features used are not described in the paper, so we extracted as many as possible from the dataset and ended up with those provided in the experimental setup. While the authors provided statistics about their dataset, our inspection of the raw dataset revealed discrepancies, as detailed in Table 1.

Furthermore, we found that 81.85% of the actual book categories were missing from the 2014 version of the metadata. The authors did not disclose any information on how they derived these numbers, or specify which exact categories were used in their research.

To address the missing category values, we supplemented the 2014 metadata by left-joining the missing categories from the more recently updated 2018 [5] and 2023 [2] versions. As a result, only 14.93% of the categories remained missing, which accounted for just 2.57% of missing category values across all interactions after merging the metadata and review data. Subsequently, we selected the top 11 most represented categories, and excluded the remaining interactions with missing category values and removed all interactions with missing values. The statistics of the final dataset used for modeling is detailed in the last row of Table 1.

3.2 Model Architecture

The model architecture follows the paper by Yu *et al.* [6], illustrated in Figure 1. The model uses a basic Two-Tower structure, generating embeddings $\mathbf{p}_u$ and $\mathbf{p}_v$ for users and items, respectively, and the similarity between these is computed via the dot product. The users and items are augmented by a vector $\mathbf{a}_u$ and $\mathbf{a}_v$, respectively, which will capture the historical user and item interactions.

In the paper, they describe that each feature $\mathbf{f}_i$ of the users and items go through an embedding layer to map them to a low-dimensional vector. This only makes sense if the feature is categorical or has no ordinal value, so for features such as the ID or

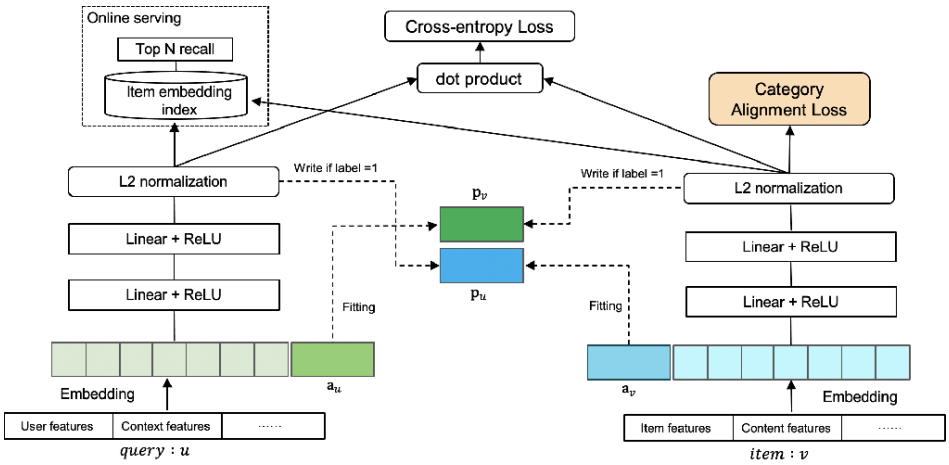

Figure 1: Network architecture of the Dual Augmented Two-tower Model [6]

the item category. However, for features such as the price or the number of helpful reviews made by a user this does not apply. So we have instead given these numerical features a fully connected layer with a dimension of 8.

Subsequently, all features are concatenated and forwarded to three fully connected layers and an L2 normalization layer. The paper sets the dimensions of the fully connected layers to 256, 128 and 32, respectively. We have changed these to dimensions of 128, 64 and 32, respectively, because the dimensions were too large for the amount of features we have, which may result in difficulty training since the model has to find an optimum in a larger space, or multiple representations of the same model.

The loss function consists of four parts: the cross-entropy loss of the dot product between the embeddings, the mimic losses for the AMM, and the CAL. The cross-entropy loss is defined as follows:

$$loss_p = -\frac{1}{T} \sum_{(u,v,y)\in\mathcal{T}} (y \log \sigma(\langle \mathbf{p}_u, \mathbf{p}_v \rangle) + (1-y) \log(1-\sigma(\langle \mathbf{p}_u, \mathbf{p}_v \rangle))) \tag{1}$$

where T is the number of user-item interactions in the training set and σ denotes the sigmoid function. The mimic losses are defined as:

$$\begin{aligned} \text{loss}_u &= \frac{1}{T} \sum_{(u,v,y)\in\mathcal{T}} \left[y\mathbf{a}_u + (1-y)\mathbf{p}_v - \mathbf{p}_v \right]^2 \\ \text{loss}_v &= \frac{1}{T} \sum_{(u,v,y)\in\mathcal{T}} \left[y\mathbf{a}_v + (1-y)\mathbf{p}_u - \mathbf{p}_u \right]^2 \end{aligned} \tag{2}$$

from which we can see that if the label is 0, so a negative interaction, the loss is 0. Otherwise, it is the squared difference between the output of one tower and the augmented vector of the other tower. Finally, the categorical alignment loss is defined as:

$$loss_{CA} = \sum_{i=2}^{n} ||\mathbf{C}(S^{major}) - \mathbf{C}(S^i)||_F^2 \tag{3}$$

where S^i is the set of items in the batch of category $i : S^i = \{\mathbf{p}_\mathbf{v}^\mathbf{i}\}$, with S^{major} the set of items of the most frequent category in the batch. C denotes the covariances and $|| \cdot ||_F^2$ is the Frobenius norm. So, minimizing this equation means minimizing the difference in covariance of the different categories with respect to the major category. When an item is in the major category, this value is 0, so the loss function penalizes items that are not in the major category. This term ensures that items in less frequent categories are only moved around in the embedding space if the improvement from other loss terms outweighs the increased CAL.

The loss functions are combined as:

$$loss = loss_p + \lambda_1 loss_u + \lambda_2 loss_v + \lambda_3 loss_{CA} \tag{4}$$

where $\lambda_1, \lambda_2, \lambda_3$ are tunable parameters. The weights of these parameters are the same as used in the paper.

We use bootstrapping to approximate the model's performance due to computational constraints. The machine learning framework used in the paper was TensorFlow, while we used PyTorch because we are more familiar with this framework. To compare the model performances, we employ the same testing methods as in the paper. This entails leaving the last interaction of every user out and evaluating the recommendations of the model with the hitrate at 50 and 100, and the MRR scaled by a factor of 10 on these interactions. The paper does not specify any hyperparameters other than the loss function and the optimizer used, so we decided the rest.

4 Experimental setup

The source code used in this project can be found here.

4.1 Data Preprocessing

During data preprocessing we created unique integer mappings for the reviewer and item identifiers. Additionally, we applied feature engineering to compute basic user statistics, such as the total number of reviews and average rating per user, and cleaned up category values that contained HTML-encoded entities, such as & with the actual & symbol to standardize them. Finally, we performed an additional filtering step to satisfy the constraint to remove all users and items with less than 5 interactions.

4.2 Model Training and Evaluation

To augment the dataset, we used negative sampling, randomly generating S negative interactions per user. Specifically, we set S to 5 and the interactions were generated by pairing the user with random items. The model was trained using the Adam optimizer, consistent with the paper, and trained 35 bootstrapped models utilizing sampling with replacement. The batch size was set at 1,536 with a learning rate of 0.05 for a maximum of 50 epochs. We used a learning rate scheduler and an early stopping function to reduce training time. The loss function follows the formulas described above, with $\lambda_1, \lambda_2, \lambda_3$ set to 0.5, 0.5 and 1, respectively.

In each bootstrapping sample, interactions were extracted from a subset of 3.5% of the total number of users. This sample was subsequently subjected to the constraints that each user and item need to have at least 5 and 4 interactions, respectively. Additionally, we ensured that the number of interactions in the sample is between 58,000 and 65,000 to minimize variability between samples. These interaction counts correspond to between 4,800 and 5,100 users. The evaluation data only contains the users and items in the sample to simulate the situation in the paper. To achieve a good estimate of the model performance, the metrics are averaged across all samples.

5 Results and Discussion

Figure 2 presents the mean loss curve over all bootstrapping samples during the training process for the model that included the CAL. Table 2 provides a detailed comparison of results, including those reported by Yu et al., as well as results for models that did not use the CAL.

The models were allowed to train for up to 50 epochs. However, models with CAL only utilized a maximum of 20 epochs, while models without CAL trained slightly longer, with a maximum of 23 epochs. This behavior suggests that the models effectively used the early stopping mechanism, which aligns with the fast convergence observed in the loss curves.

Overall, the results we obtained are consistent with those reported by Yu *et al.* While the hit rates (HR) were slightly lower, they remain within the same order of magnitude. This indicates that the applied bootstrapping methodology successfully reproduced their findings. Interestingly, despite slightly lower hit rates compared to Yu *et al.*, our model showed an improvement in the Mean Reciprocal Rank (MRR). This discrepancy is likely not due to an improved model but rather a consequence of how MRR is influenced by dataset size. Items predicted poorly by the model are ranked later in the recommendations, leading to lower reciprocal ranks. Since the bootstrapping samples are much smaller than the full dataset, this effect positively skews MRR results.

A notable observation is the very low standard deviations in the loss metrics, indicating consistent performance across the 35 bootstrapping samples. This consistency reflects that the bootstrapping methodology effectively produced representative samples of the entire dataset. Consequently, the results can be considered a reliable estimate of the model's performance on the full dataset.

Yu *et al.* demonstrated the effectiveness of CAL, reporting an increase across all performance metrics. Our study observed similar results, further supporting their claims regarding the benefits of CAL for model performance.

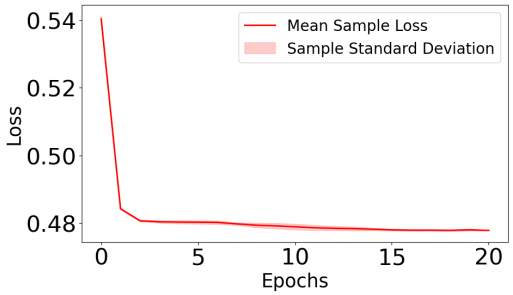

Figure 2: Mean loss curve of bootstrapped DAT using the CAL

Table 2: Performance comparison between the results from Yu *et al.* and our experiments

Models	HR@50	HR@100	MRR
DAT (Yu *et al.*)	**0.0519**	**0.0836**	0.0711
DAT	0.0470	0.0739	**0.0837**
	± 0.0066	± 0.0088	± **0.0143**
DAT(w/o CAL) (Yu *et al.*)	0.0503	0.0816	0.0698
DAT(w/o CAL)	0.0459	0.0712	0.0741
	± 0.0145	± 0.0182	± 0.0228

6 Conclusion

In this paper, we used bootstrapping to estimate the performance of the Dual Augmented Two-tower model proposed by Yu *et al.* to test if we can use a lightweight approach to estimate model performance. The results we achieved were similar, though slightly worse, to those from Yu *et al.* We can therefore conclude that our implementation of bootstrapping gives a good estimation of model performance, with the caveat that you have to be careful what influence this has on the evaluation metrics used. Furthermore, our results also support the claim that the CAL improves the model's performance.

References

[1] Yuhang Cui, Shengbin Liang, and YuYing Zhang. 2024. Multimodal representation learning for tourism recommendation with two-tower architecture. *PLOS ONE* 19, 2 (Feb. 2024), e0299370. https://doi.org/10.1371/journal.pone.0299370 Publisher: Public Library of Science.

[2] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging Language and Items for Retrieval and Recommendation. *arXiv preprint arXiv:2403.03952* (2024).

[3] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, San Francisco California USA, 2333–2338. https://doi.org/10.1145/2505515.2505665

[4] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. https://doi.org/10.48550/arXiv.1506.04757 arXiv:1506.04757.

[5] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 188–197. https://doi.org/10.18653/v1/D19-1018

[6] Yantao Yu, Weipeng Wang, Zhoutian Feng, and Daiyue Xue. 2021. A dual augmented two-tower model for online large-scale recommendation. *DLP-KDD* (2021).