Recommending Electronic Items on Amazon Using Dynamic Temporal Embeddings

Course project for CSE 6240: Web Search and Text Mining, Spring 2022

Ahindrila Saha
asaha70@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Akshay Jadiya
ajadiya3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Harshal Gajjar
hgajjar3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Yashwant Singh
ysingh64@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

Recommendation systems are widely used in e-commerce and are crucial for a platform's success because they can help in reducing transaction costs of finding and selecting items, increase sales as a result of personalized offers and improve user experience. In this project, we are recommending electronic items on Amazon using temporal dynamic embeddings. Existing recommendation systems are static and are insufficient to model the temporal evolution of the users and items. So, we use new state of the art algorithms such as JODIE and CTDNE to generate embedding trajectories of users and items using rating, and review data from Amazon to predict future items. The dataset which we are using consists of interactions of users with items along with their ratings and reviews. The dataset has 52,621 user ratings (and reviews), with 7,803 unique users and 998 unique items. The performance of both the models is compared using Recall@10 and MRR (mean reciprocal rank). Our aim is to predict the item user will interact with next based on the given history of user-item interactions.

We achieve Recall@10 of 0.299 and MRR of 0.197 for CTDNE and Recall@10 of 0.97 and MRR of 0.92 for the JODIE approach. JODIE is preferred over CTDNE as it is more scalable and can provide embeddings for every point in time which is also visible by the results that we get.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Continuous-Time Dynamic Network Embeddings, Recurrent Neural Networks

ACM Reference Format:

Ahindrila Saha, Akshay Jadiya, Harshal Gajjar, and Yashwant Singh. 2018. Recommending Electronic Items on Amazon Using Dynamic Temporal Embeddings: Course project for CSE 6240: Web Search and Text Mining, Spring 2022. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The objective of this report is to generate recommendations of electronic items using Amazon review dataset. Temporal interaction networks provide a good way to represent the time-evolving and dynamic interactions between entities. Representation learning provides a tool to model the temporal evolution of items and users where each item and user can be embedded in a Euclidian space. However, interests of the users and the items constantly evolve with time, so a single static embedding is not sufficient as the recommendation becomes stale. Real networks also have millions of nodes and edges so the models used should be scalable and able to operate on large datasets. Also, to make recommendations practically useful, the model should be able to provide predictions in near constant time irrespective of the number of items. Our original dataset has 7.8 million rows. Due to resource constraints, we filtered the dataset for only top 1000 items and 8000 users. Our preprocessed dataset has 52621 rows which includes 998 unique items and 7803 unique users. For CTDNE, we used time as edge features and created embeddings of length = 128 whereas for JODIE, we used ratings as edge features and embeddings of length = 4.

So, we use JODIE to generate embedding trajectories of users and items using their interactions as they are dynamic and provide predictions in real time. We compare results with CTDNE which can generate future embeddings. We prefer JODIE over CTDNE as it is more scalable and can generate embeddings for every point in time.

The predictions can be used to provide recommendations using distance in the embedding space. The recommendations can improve personalized user experience. Therefore, these methods can potentially be used by any e-commerce platform in order to increase revenue by improving user engagement. We are able to achieve Recall@10 0.299 and MRR 0.197 for CTDNE and Recall@10 0.97 and MRR 0.92 for the JODIE approach on the validation dataset. We also give the top 10 recommendations for each user using CTDNE.

2 LITERATURE SURVEY

Amazon uses its proprietary A10 algorithm to recommend products to customers. The algorithm considers a user's shopping and browsing history which is then used to match the data against a multitude of factors regarding products including product rank, keyword relevancy, listing traffic, customer reviews, average star ratings, etc.

Given the algorithm is proprietary, we could not figure out what the limitations are. However, one thing that we did notice is that sometimes the algorithm ends up recommending "out of stock" items which could be annoying to users.

Brent et. al. in 2017 [3] talk about improvements and adaptations for item-based collaborative filtering for the E-commerce website Amazon.com. The algorithm finds related items for each item in the catalog and items that are purchased with an unusually high frequency by people who bought some item are used for generating recommendations for other users. To better understand the human behavior and thus recommend items, it is important that we understand how the user needs change through time. Unfortunately, collaborative filtering (item based as well) does not take time into account. Additionally, collaborative filtering does not take item-sequence into account. In our project, we have majorly focusing on time dynamic models.

Nguyen et. al. in 2018 [2] propose methods for learning time-respecting embeddings from continuous-time dynamic networks. They describe a framework which serves as a basis for incorporating temporal dependencies into existing node embedding and deep graph models based on random walks. This approach learns appropriate network representation from continuous-time dynamic networks that captures the relevant temporal dependencies of the networks at the finest most natural granularity. They are able to achieve average gains for AUC scores of 12% over other methods (DeepWalk, Node2Vec, LINE). We are using CTDNE just to generate embeddings and then using those to find most similar items for a user. Typically CTDNE embeddings are used for the link prediction task, however, here we are using cosine similarity to generate recommendations. Another difference in our approach is that we used CTDNE for generating embeddings for heterogeneous nodes (user and item). Typical examples show implementation of CTDNE in homogeneous graphs only. Our dataset includes userId/ product id, rating, reviews and time of rating which fulfills the criteria for the type of data required to use CTDNE framework. This method has the limitation of generating embeddings for every point in time and to circumvent this, we tried another method (JODIE) which is summarized below.

Srijan et. al. in 2019 [1] proposed JODIE, a coupled recurrent neural network model that learns embedding trajectories of users and items. It also models the future embedding trajectory of a user and item which can be used to estimate the embedding of the user at any time in the future. A t-Batch algorithm is developed to make the method scalable. The JODIE approach outperforms six state-of-the-art algorithms on four datasets in predicting user state change and future interactions. One of the issues with JODIE is that learning embeddings for individual users/items is expensive - this can be solved by learning trajectories for groups of users/items. Another limitation is that JODIE does not consider sessions. Given our dataset contains userid, product id, rating, time of rating, we plan to use the JODIE approach to estimate future user/item embeddings and recommend items accordingly. JODIE has one layer in which the output dimension is equal to the number of items in dataset. This creates a limitation on the number of items that a dataset can contain based on the available VRAM (which is typically smaller than RAM.) JODIE also uses simple RNN rather than state-of-the-art sequence models like transformers. Finally, an innovative direction would be to design new items based on missing predicted items that many users are likely to interact with.

3 DATASET DESCRIPTION

3.1 Preprocessing

We are using a subset of *Amazon product data* from *Repository of Recommender Systems Datasets* hosted by UCSD. The original dataset contains 7,824,482 user ratings (and reviews), with 4,201,696 unique users and 4,76,002 unique items. However, running JODIE with such a high number of items was not possible even on a machine with 32GB of VRAM (because of the large number of items.) We observed that the all the datasets (Reddit, Wikipedia, MOOC and LastFM) that the JODIE authors used to compare it with other models had significantly lower number of items. So, we selected 1000 most rated items and then filtered 8k users who had given the highest number of ratings. These numbers were chosen to match that of Wikipedia dataset. In addition to that, we also subtracted minimum timestamp from the dataset from all timestamps to make our time start from 0. The exact steps executed are listed below:

1. Each item and user were replaced by a unique index starting from 0.
2. The dataset was filtered for top 1k items and 8k users based on the total number of interactions of users and items. After these steps, the number of rows reduced to 52,621.
3. The timestamp column was adjusted by subtracting the minimum value of all the timestamps from each row to make all the timestamps start from 0.
4. Data was sorted according to the timestamp.

3.2 Raw Data Statistics

Our subset contains 52,621 datapoints with an average rating of 4.4 and with timestamps range from 28 April 2000 to 21 July 2014. Detailed statistics about the data are given in Table 1, Table 2, Figure 1, Figure 2, Figure 3 and Figure 4.

4 EXPERIMENT SETTING

We are trying to predict the electronic items that users will interact with, given their history of interaction with similar products. We

Measure	Value
Number of unique users	7803
Range of number of ratings per user	{1,79}
Mean number of ratings per user	6.74369
Median number of ratings per user	6.

Table 1: Dataset Users Overview

Measure	Value
Number of unique items	998
Range of number of ratings per item	{1,611}
Mean number of ratings per item	52.7265
Median number of ratings per item	36.5

Table 2: Dataset Items Overview

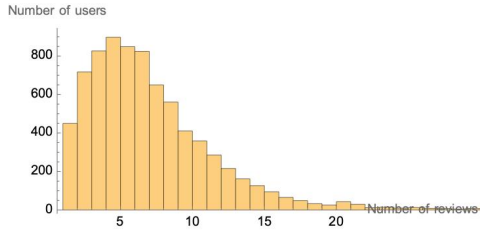

Figure 1: Number of users for a given number of ratings in the Dataset

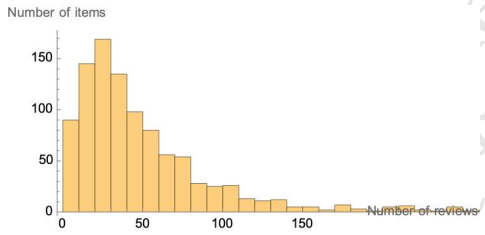

Figure 2: Number of items for a given number of ratings in the Dataset

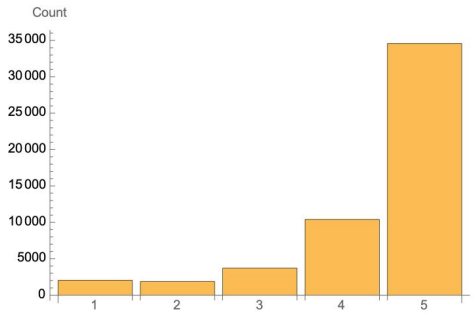

Figure 3: Rating distribution in the Dataset

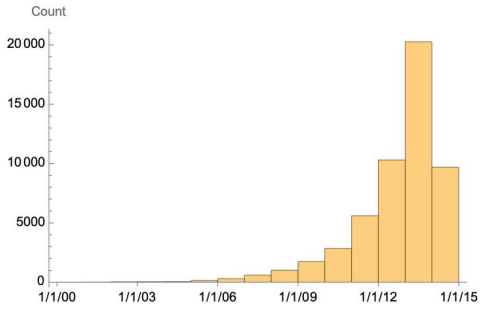

Figure 4: Date histogram of rating timestamps in the Dataset

plan to use Recall@10 and Mean Reciprocal Rank (MRR) as our evaluation metrics. Since this is the application of recommendation systems in an e-commerce setting, the rationale behind choosing Recall@10 is that we want to see how many relevant items are being recommended to the user after putting the search query. And the rationale supporting MRR is to figure out how many items must be recommended to get the correct recommendation. More relevant items will lead to increased likelihood of purchasing the product, which in turn will increase the revenue.

4.1 System specifications

System specifications of the machine used have been defined in table 3.

Parameter	Value
CPU	Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz
Disk	80GB
Memory	187GB
GPU	Tesla V100
vRAM	32GB

Table 3: System specifications

4.2 Dataset Split

We split the database by time, i.e. considering fixed 2 timestamps t_1 = Dec 21, 2013 10pm GMT-4 and t_2 = Mar 16, 2014 10pm GMT-4 where $t_1 < t_2$ are such that every interaction before t_1 was used as training data, every interaction between t_1 and t_2 was be used for validation and every interaction after t_2 was be used for testing. The values t_1 and t_2 are such that the Table 4 values hold true.

Train	Val	Test
80%	10%	10%

Table 4: Dataset split

5 METHOD

The CTDNE paper focuses on learning time-respecting embeddings for continuous-time dynamic. Given a temporal network G, the

goal is to learn a function that maps nodes in G to D-dimensional time-dependent feature representations suitable for a down-stream machine learning tasks.

The CTDNE paper suggests sampling a set of "temporal" random walks from all possible random walks. A temporal random walk very similar to a standard random walk with a constraint that edges must be traversed in increasing order of edge times. The embedding learning algorithm optimizes the log likelihood of a temporal context window when observed w.r.t. a node. The temporal context window defines a walk as valid only when its length is between ω (min length) and L (max length). A temporal context window count β is defined as the total number of context windows of size ω that can be derived from the set of temporal random walks. The inputs given to the model are a (un)weighted and (un)directed dynamic network G, temporal context window count β, context window size ω, embedding dimensions.

JODIE is a mutually coupled Recurrent Neural Network model that learns to generate embedding trajectories of users and items from their interactions. JODIE trains two RNN'S one for user and the other for items which update the embeddings of user and items using each other embeddings. JODIE trains a projection operator that can accurately predict the embedding trajectory at any future point in time. JODIE can then use the embeddings to predict the next item that the user is likely to interact with. The model is trained in such a way that the embedding of the predicted item is equal to the next real item. Using the list of predictions, we can rank items using distance in the embedding space to generate recommendations in near constant time. To make JODIE scalable to large networks, T batching is used. A sequence of batches is created such that all the interactions in a batch can be processed in parallel and processing the batches in sequence maintains the temporal dependency of the users and items. This makes JODIE scalable to large number of interactions. The link for the repository can be found here. For our instance of JODIE, we used embedding dimension of 4, and trained it for 30 epochs while keeping all other hyper parameters to their default faults.

The reason we chose JODIE and CTDNE is because both methods allow us to generate dynamic embeddings of the users and items which can be used to generate recommendations over time. CTDNE and JODIE both can generate interdependent user and item embeddings and can be trained in batches. However, JODIE is superior because it allows us to generate embeddings at any point in time.

6 EXPERIMENTS AND RESULTS

We are trying to predict the electronic items that users will interact with, given their history of interaction with similar products. We plan to use Recall@10 and Mean Reciprocal Rank (MRR) as our evaluation metrics. Since this is the application of recommendation systems in an e-commerce setting, the rationale behind choosing Recall@10 is that we want to see how much relevant items are being recommended to the user after putting the search query. And the rationale supporting MRR is to figure out how many items must be recommended to get the correct recommendation. More relevant items will lead to increased likelihood of purchasing the product, which in turn will increase the revenue.

6.1 CTDNE Experimental Setup

The filtered dataset consisted of 52k interactions between 8k users and 1k items as explained in the data cleaning section. The dataset having user id, item id and the timestamp associated with the interaction was used to create a graph using StellarGraph library (link - https://stellargraph.readthedocs.io/en/stable/index.html). The StellarGraph library offers state-of-the-art algorithms for graph machine learning, making it easy to discover patterns and answer questions about graph-structured data. The time of the interaction was used as edge weights of the StellarGraph object.

One of the defining features of CTDNE which differentiates it from other static embedding generating algorithms is that it generates a "temporal random walk" (more details in the CTDNE method overview section). Using StellarGraph's temporal random walk generation function, we generated 1.1 million random walks using the following hyperparameters:

Parameter	Value
Number of walks per node	10
Embedding dimension	128
Maximum walk length	80
Minimum walk length	70
Context Window Size	10

Table 5: CTDNE hyperparameters

After generating random walks, we generated embeddings of length 128 using Gensim's Word2Vec function. This way we were able to generate 8631 embeddings (7803 for the use nodes and 998 for the item nodes). The user and item embeddings were separated intro different dictionaries and then cosine similarity was calculated for each pair of user and item.

Using the cosine similarity metric, we calculated top 10 recommendations for each user by taking products with 10 highest cosine similarity scores. Using these recommendations, the following were the values of recall@10 and MRR for the test dataset:

Metric	Score
Recall@10	0.299
Mean Reciprocal Rank	0.197

Table 6: CTDNE results

6.2 JODIE Experimental Setup

The filtered dataset consisted of 52k interactions between 8k users and 1k items as explained in the data cleaning section. The networks are stored under the data folder, one file per network. The network is created from the dataset in the following format:

- One line per interaction/edge.
- Each line should be user, item, timestamp, state label, comma-separated array of features.
- First line is the network format.
- Timestamp was already a cardinal format
- State label should be 1 whenever the user state changes, 0 otherwise. As there was no state labels, it was kept as 0

- Feature list was created from the rating column

The authors of JODIE have conveniently uploaded their code to GitHub, which we used for our project
(link - https://github.com/srijankr/jodie)

We used PACE with a GPU cluster for running the code. Recent versions of PyTorch, numpy, sklearn, tqdm, and gpustat were installed. We created directories to store data and outputs. All the models and necessary codes were saved in the folder. We trained the JODIE model on the electronics network using the code jodie.py file. The following hyper parameters was used:

Parameter	Value
Epoch (with early stopping)	30
T-batch Timespan	10.4 days
User embeddings dimension	4
Item embeddings dimension	4

Table 7: JODIE Hyperparameters

We evaluated the performance of the model in predicting interactions using Recall@10 and MRR on the test dataset.

Metric	Score
Recall@10	0.97
Mean Reciprocal Rank	0.92

Table 8: JODIE results

6.3 Results

- Being a deep learning based model, JODIE outperformed CTDNE which is a random walk based method because Deep learning can efficiently learn the underlying explanatory factors and useful representations from input data and Deep learning is powerful for sequential modeling tasks.
- JODIE has a projection component which can be used to generate future embeddings and consequently to give recommendations
- CTDNE lacks such a functionality and hence, we had to use embeddings from the training set for generating recommendations
- Due to these two differences, we see a difference in the performance of the two models

7 CONCLUSION

7.1 Shortcomings

JODIE and CTDNE both do not take into account sessions of the user. As a result, both methods take the whole interaction history of the user to generate recommendations and some of them might not be relevant. Therefore, applying a method like LatentCross which is used to generate YouTube recommendations may help generate better session dependent recommendations. A helpful addition to the current model would be to include "context" such that the model can generate different recommendations based on user state, for example, recommendations generated after a product search should be different from recommendations shown on the home page after a fresh log in. One shortcoming comes from out sampling of the original dataset to reduce its size. We introduce popularity bias since we only consider the most popular 1k items.

7.2 Extensions

As our dataset only contained ratings and timestamp which we used as features for our models. In order to generate more accurate recommendations, user demographic data or item details can be used as features while training to generate more robust recommendations. Item details and user preferences can be extracted from the corresponding reviews data using NLP techniques, this will also lead to a better item representation and therefore better recommendations.

8 CONTRIBUTION

All team members have contributed a similar amount of effort.

REFERENCES

[1] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1269–1278. https://doi.org/10.1145/3292500.3330895

[2] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 969–976. https://doi.org/10.1145/3184558.3191526

[3] Brent Smith and Greg Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* 21, 3 (2017), 12–18. https://doi.org/10.1109/MIC.2017.72