amazon Electronics Recommendation

CSE 6240 - Web Search & Text Mining - Spring 2022

Team

Harshal Gajjar
Akshay Jadiya
Ahindrila Saha
Yashwant Singh

Problem Description

Given the previous interactions $(rating, time, item, user)$ with the electronic items listed on Amazon, can we predict which item the user will rate next? How well is the prediction?

Motivation and Importance

Good recommendations are crucial for an e-commerce platform's success, because of several reasons:

- **Reduce transaction costs** of finding and selecting items in an online shopping environment
- **Increase sales as a result of very personalized offers**
- **Improves overall user experience** of the platform as the time to achieve their goals is reduced

Existing Method 1 - CTDNE

CTDNE stands for Continuous Time Dynamic Network Embeddings.

It is quite like DeepWalk except instead of random walks it **considers temporal random walks**.

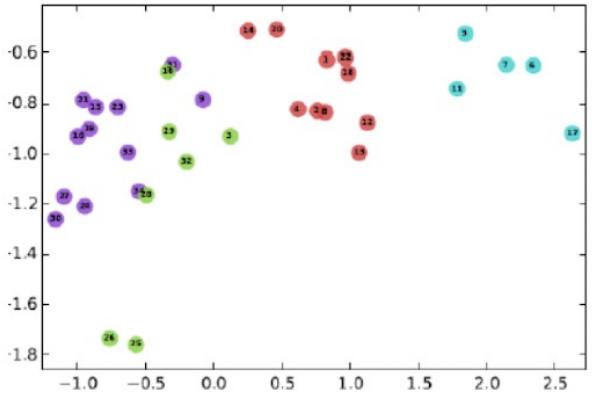
> *A temporal random walk very similar to a standard random walk with a constraint that edges must be traversed in increasing order of edge times.*

In case of CTDNE, the **walks are valid only if their length is between ω (min length) and L (max length)**. These are two hyperparameters for the model.

Goal of CTDNE is to learn embeddings such that:

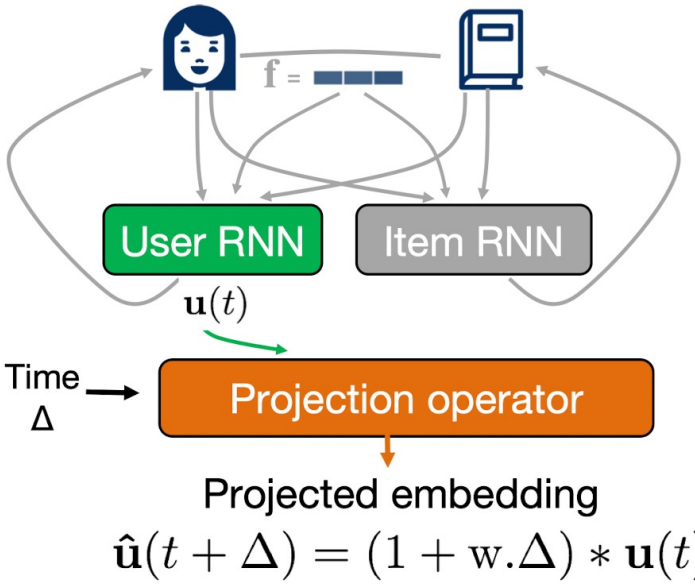
1. **The similarity of embeddings is high for node pairs co-appearing in random walk**
2. Low probability for other node pairs that do not co-appear in random walks

Input



Output

Existing Method 2 - JODIE



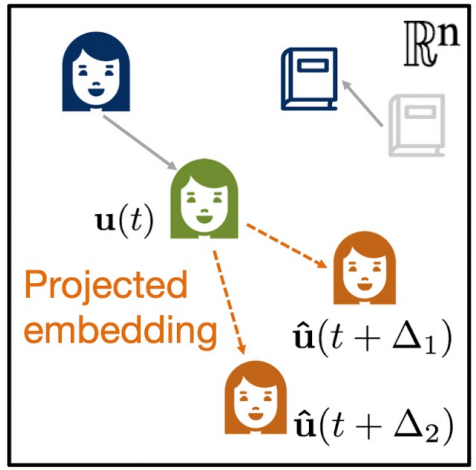
JODIE is a **mutually recursive and coupled Recurrent Neural Network** framework.

It **trains two RNNs one for user and the other for items** which update the embeddings of user and items using each others' embeddings.

It **also trains a projection operator that can accurately predict the embedding trajectory** at any future point in time. This trajectory is used to find the items that users will interact with.

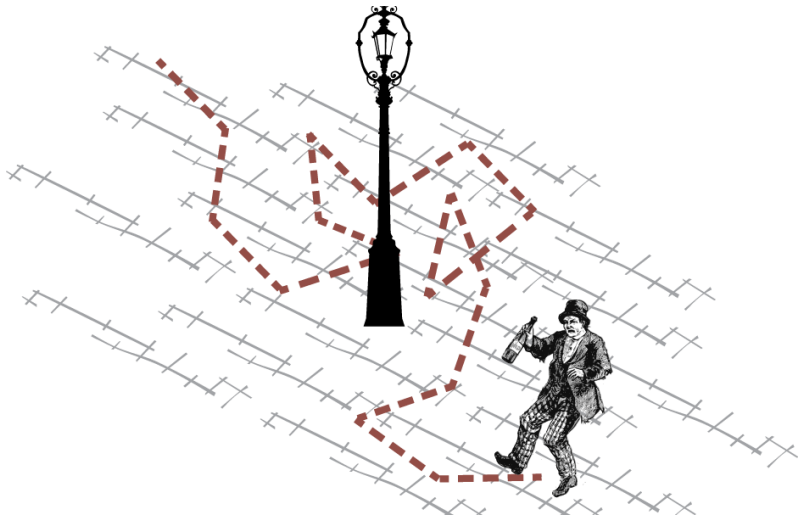
To make JODIE scalable to large networks, T-batching is used. In T-batching the generated batches in maintain the temporal dependency of the users and items.

Goal of JODIE is to train model such that:
Embedding of the predicted item ≈ **Embedding of the real next item**, where predicted item is generated from a linear layer executed on user embedding.

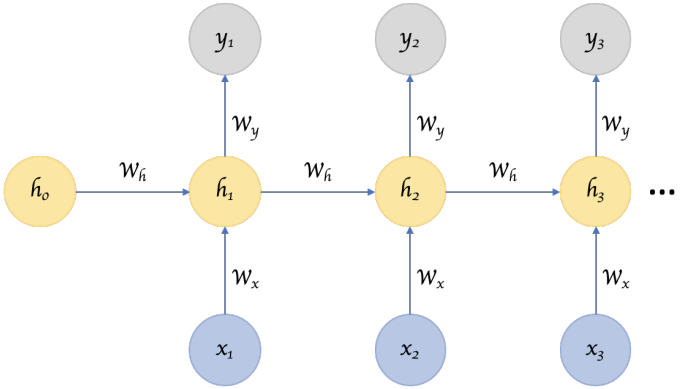


Why these models?

We were curious to see how much of a difference there would be between a good random walk-based method, i.e. CTDNE, and a good neural network-based method, i.e. JODIE. Both the models do consider temporal aspect of the data.

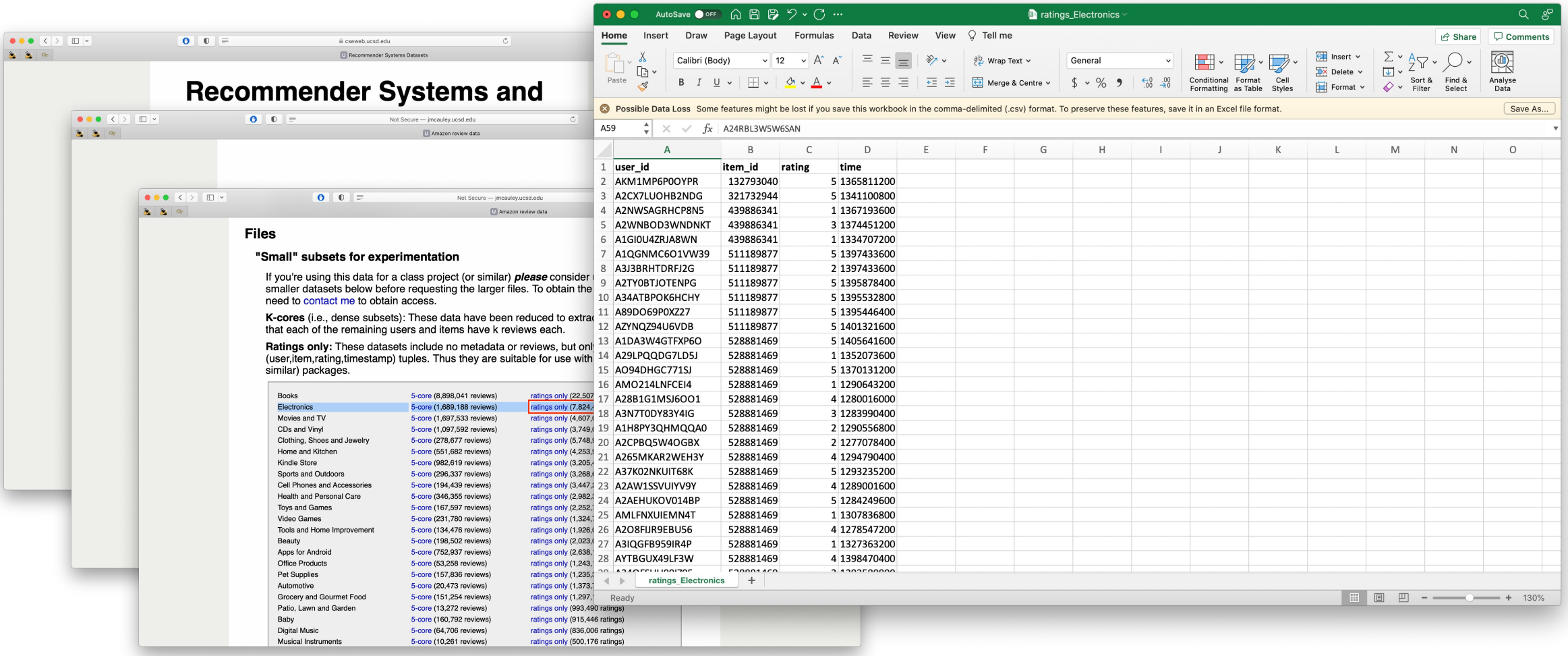


versus



Dataset Details

1. Obtaining data



Dataset Details

Reduced dataset ⟩

2. Cleaning data

We, firstly **reduced the number of items to 1000 by selecting the most popular ones** and then the number of **users to 8000** by selecting the most active ones. These numbers are like that of the Wikipedia dataset (that the authors used for benchmarking.)

This constraint was due to hardware resource, as JODIE's last layer generates a vector with dimension equal to the number of items in the dataset. The original dataset ran out 32GB GPU out of memory.

After the reduction, **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.

> Exact cleanup steps:
> 1. Each **item and user were replaced by a unique index** (0-maximum of user or item)
> 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
> 4. **Data was sorted according to the timestamp**

Dataset Details

3. Data properties

Users

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

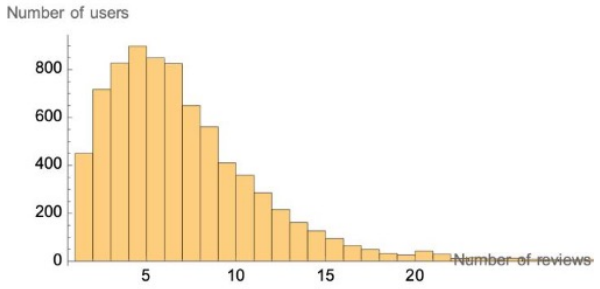


Figure 1: Number of users for a given number of ratings in the Dataset

Items

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

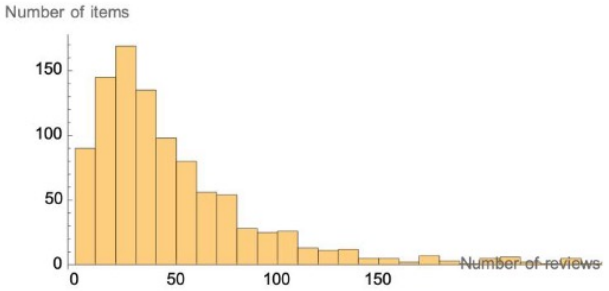


Figure 2: Number of items for a given number of ratings in the Dataset

Ratings

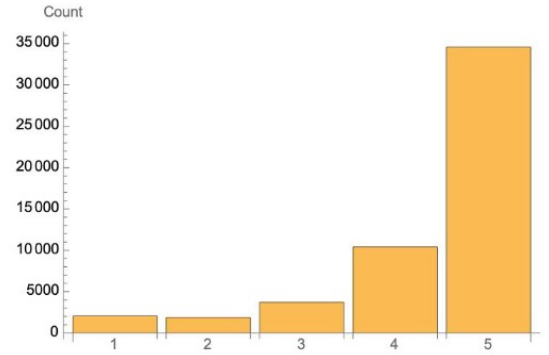


Figure 3: Rating distribution in the Dataset

Time

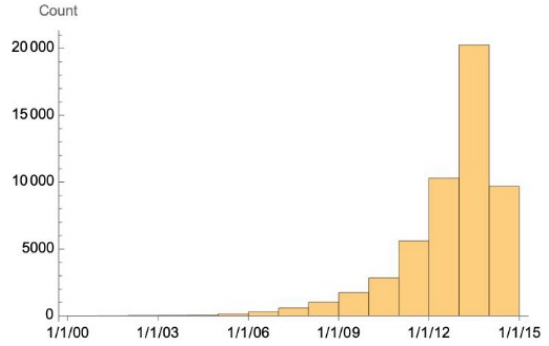


Figure 4: Date histogram of rating timestamps in the Dataset

Experimental Setup

We used PACE for all our code execution, with the system specifications being as following:

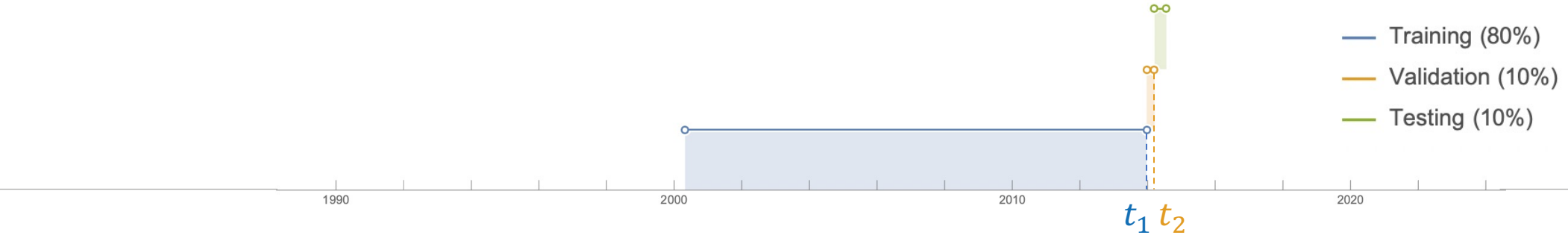
Parameter	Value
CPU	Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz
Disk	80GB
Memory	187GB
GPU	Tesla V100
vRAM	32GB

Table 3: System specifications

We split the database by time, i.e. considering fixed 2 timestamps:

= Dec 21, 2013, 10pm GMT-4

t_2 = Mar 16, 2014, 10pm GMT-4



Experimental Setup

In our approach, **we predict the electronic items that users will interact with, given their history of interaction with similar products**.

We used **Recall@10 and Mean Reciprocal Rank (MRR)** as our evaluation metrics.

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.

The rationale supporting MRR is to figure out how **many items must be recommended to get the correct recommendation**.

Results

CTDNE	JODIE

Metric	Score
Recall@10	0.299
Mean Reciprocal Rank	0.197

Metric	Score
Recall@10	0.97
Mean Reciprocal Rank	0.92

Takeaways

- **JODIE** (neural network based) **outperformed CTDNE** (random walk based)
- 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.

Limitations & Future Work

- Both **JODIE and CTDNE do not consider 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. **A better model can be developed/used to further improve the scores.**

- **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. This issue can be perhaps solved by **learning embeddings for a group of items rather than each.**

- **A helpful addition to the current model would be to include "context"** such that the model can generate different recommendations based on user state. JODIE does support this, but we did not have data for this.
    - For example, recommendations generated after a product search should be different from recommendations shown on the home page after a fresh log in.

Thank you