*Page-Level Optimization of e-Commerce Item Recommendations*

*Devinesh Singh  
Department of Computer Science  
San Jose State University*

*San Jose, United States  
devinesh.singh@sjsu.edu*

*Abstract* Recommendation Systems attempt to predict user preference and user ratings of an item; we see a recommendation system on nearly all e-commerce website, usually below an item description. This list usually contains a list of items that are similar to products that a customer viewed and/or searched during their web session; the issue with this list is that the order that the products appear is fundamental to user engagement: if the products are not ordered according to relevance, a customer will likely not show interest in the products and the business may lose out on potential revenue. Neural Networks are algorithms designed to capture relationships between data. As such, a Neural Network can be applied to the recommendation problem to make personalized recommendations to a user based on the user’s web history.

Keywords—e-Commerce, Recommendation Systems, CNN-LSTM

# Introduction

Recommendation Systems have become ubiquitous and unavoidable in the last few decades with the rise of web services and e-commerce; these systems that were designed to optimize consumer choices have been implemented in practically all online platforms – Netflix, Amazon, Apple App Store, and Target. Additionally, with the rise of e-commerce, especially during the cov-2 pandemic, consumers face the issue with information overload – availability of excess information aimed to complete a task; particularly, unlike traditional retail, where the business must be selective in the products chosen to display due to scarcity of shelf space, e-commerce does not necessarily have a finite threshold on the number of items that can be displayed on a website. With a near-infinite number of items available to consumers, the need for a recommendation system – algorithm used to filter through millions of products and suggest a list of relevant products to its users based on user behavior – is necessary to avoid information overload. Not only that, but the recommendation system should also curate a unique list of products to each user, based on the user’s past behavior.

# Related Works

Collaborative Filtering (CF) is one of three main types of recommendation systems; it can be broken into two sub-categories: item-based and user-based systems. Vijay et. al. proposed to use K-Nearest Neighbors (KNN), alongside user-based CF to curate a list of items for users; KNN is a supervised learning technique used to find the top k-items based on common item ratings and make predictions for a user using the average rating of the top k-nearest neighbors. A user-based system finds k-users who have similar preferences as the target user and computes the average of the ratings of the k-users; the average rating is used as the target user’s rating of the item. Top-K rated items are used as the list of recommended products for the target user. Liao et. al. used an item-based system to recommend items to users; item-based systems find items of k-users with similar preferences as the target user. Cosine Similarity is used to determine the rating of all item pairs and a weighted sum of the ratings is taken to determine a list of recommendations.

Luo et. al. proposed to use a content-based system to use actions of a user to recommend products to a user. Particularly, the system builds a user profile to exploit a users’ historical behavior to recommend similar products; for example, if a user likes comedy movies, the system recommends more movies within the comedy genre and fewer movies within the horror genre.

Furthermore, Rocca et. al. proposed to use deep neural networks (DNN) to address limitations of the matrix factorization-based methods mentioned above; particularly, unlike matrix factorization methods, DNNs can incorporate side information with the item and user features to capture specific interests of a user and improve the relevance of recommendations.

# Baseline paper

## Original Baseline

Ilardi et. al collected their dataset through Thompson Sampling – a heuristic algorithm used to overcome the explore-exploit dilemma – and session-based user-intent attribution; particularly, the authors gathered implicit user signals – whether a user clicked on an item, added an item to their cart, and/or searched for an item – to determine positive and negative labels for the data. A module is defined to be a positive label if a user had interacted with any item within a page (i.e. user clicked on an item and/or purchased an item) and negative if a user did not interact with any item within a page.

Ilardi et. al. used the extract, transfer, and load (ETL) processes to integrate the different sources of information gathered and build the training dataset. Personalized results are accomplished through the integration of a user’s search, view and purchase history with module interactions; the user engagement signals are aggregated at different granularities and used as features in the model. Features are normalized to ensure the values are restricted to a small range of numbers.

Ilardi et. al proposed a two-stage network built upon a Recurrent Neural Network (RNN) that utilizes long short-term memory (LSTM). Users and items are embedded into d-dimensional vectors; the embeddings are concatenated so the LSTM can learn dependencies and relationships between the user and item. LSTM cells extract useful information from the embeddings that can be used to predict a user rating towards an item. Multilayer Perceptron (MLP) layers are modified to consider multiple user engagement signals.

Beam Search – a heuristic that selects the top k-scores – is used to find and ensure an optimal sequence of products is presented to each customer and ordered according to overall relevance to the user’s shopping intent. At each time step, beam search maintains a sequence of items with the highest scores.

Several evaluation metrics were used to measure the performance of the model; particularly, F1 Score, Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Purchase Rank. Ilardi et. al. focused primarily on F1 Score.

PyTorch libraries were used to implement the framework for the model. Adam Optimizer – a stochastic gradient descent method – with a 0.001 learning was chosen as the hyperparameters of the LSTM model. 1 million pages were split into 80/10/10 training, validation, and testing sets respectively. Embedding dimensions were set to 50 and the number of linear layers were configured between 1 to 3. User Engagement Signals also varied in each experiment to consider the effect of personalized features to the model’s performance.

## Revised Baseline

Ilardi et. al. did not provide the source code nor dataset used throughout the paper; additionally, the paper only briefly describes the implementation details of the LSTM, Thompson Sampling, and Beam Search. Nevertheless, source code provided by Carta et. al. constructed a LSTM model, similar to Ilardi et. al. with the exception that it was used to recommend movies using the MovieLens dataset. Few modifications were necessary to get the model to work correctly with the RocketRetail dataset.

RetailRocket dataset was used in lieu of the dataset Ilardi et. al. used throughout their research. RetailRocket contains 1.41 million unique customers, 467,000 unique items, and 2.756 million events: user view of item, user purchase of item, and item added to cart. Unfortunately, the RetailRocket dataset did not contain a list of items searched by a user in their web session and cannot be used to provide personalized results at page-level as Ilardi et. al.

RetailRocket was pre-processed by the author of the dataset: duplicates and entries with missing fields were removed from the dataset. Author initially split the dataset into three files; Excel’s combine and load option was used to merge the files using attributes that matched in the three files. Additionally, the property and value fields were dropped from the dataset as they did not contribute to the overall prediction of the model.

Carta et. al. opted for a single LSTM layer, whereas Ilardi et. al. configured the LSTM layers to vary between 1 to 3. Ilardi et. al did not provide details pertaining to which of the models the results correspond to, hence a single LSTM layer was used. Furthermore, beam search was removed from the model as it resulted in sub-par results. Additionally, the model embeddings needed to be changed as Carta et. al. sought to recommend a list of movies, whereas we are tasked to curate a list of products from e-commerce websites.

Hyperparameter Tuning was performed on the learning rate to determine the value that converged to the optimal solution. Additionally, the model ran for 100 epochs, instead of the 10 epochs proposed by Ilardi et. al to ensure the model converges.

# Proposed Solution

## CNN-LSTM Architecture

## CNN-LSTM architecture is a hybrid neural network designed for sequential prediction problems; the architecture combines convolutional neural network layers for feature extraction on the input data and LSTM layers to perform sequential prediction on the feature vectors. Input is fed through the convolutional layers to extract useful information that can be used to predict user ratings of a product. LSTM then takes the extracted user and item features to predict user rating and preference of products. Recommended products are sorted based on highest scores.

CNN-LSTM model followed a similar procedure as the architecture implemented by Carta et. al. RetailRocket dataset was used through all experiments. Adam Optimizer and 100 epochs were used. Additionally, the number of hidden layers and learning rate were tuned accordingly.

Initially, the proposed model consisted of one convolutional layer and one LSTM layer; however, the basic CNN-LSTM model performed poorly. Hence, an additional convolutional layer was added to the model to increase its depth and ability to extract higher-level features from the input data.

## BERT Transformers

Iladi et. al. planned to evaluate transformer-based optimization models against the RNN recommendation in future works. Hence, BERT was implemented to compare against the RNN-based recommendation system and the CNN-LSTM recommendation. BERT4Rec – a network trained to predict movies from a user's history – was used to predict user ratings of items.

# Evaluation

EVALUATION OF MODELS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Learning Rate | Hidden Layer Size | F1 Score | Runtime (ms) |
| Ilardi Baseline | 0.001 | N/A | 0.56 | N/A |
| Carta (Revised) Baseline | 0.01 | 128 | 0.1378 ∓0.01341 | 27831.1 |
| Carta (Revised) Baseline | 0.1 | 128 | 0.4980 ∓0.01242 | 33291.3 |
| Carta (Revised) Baseline | 0.1 | 256 | 0.5786 ∓0.00134 | 40193.4 |
| CNN-LSTM (1 Conv Layer) | 0.01 | 128 | 0.2302 ∓0.00214 | 27831.1 |
| CNN-LSTM (1 Conv Layer) | 0.1 | 128 | 0.2423 ∓0.00324 | 30007.3 |
| CNN-LSTM (2 Conv Layers) | 0.1 | 128 | 0.9313 ∓0.00301 | 42937.9 |

# Conclusion

Unfortunately, the BERT Transformer architecture did not provide insightful results; the BERT transformer constantly recommended products that did not appear in the RetailRocket dataset, resulting in an F1 score near 0. Although transformer architectures have been proven to perform better in natural language processing (NLP) use cases, the transformer was not applicable in the case of recommendation systems. BERT transformers may need further attention to correctly configure the model to properly recommend products to consumers. Upon further review, I could not pinpoint the reason the BERT Transformer had malfunctioned; however, this is something that could be investigated during summer break.

CNN-LSTM performed significantly worse than LSTM when only a single convolutional layer was used; however, the addition of a second convolution layer dramatically increased the overall performance of the model. Generally, a lower learning rate and hidden layer size led to poor results; as the learning rate incrementally increased from 0.001 to 0.1, the performance of models increased. Hidden layer size of 256 increased the performance of the LSTM by nearly 2x.

Two-layered CNN-LSTM significantly outperformed both Ilardi et. al. baseline model and the revised LSTM model by Carta et. al. F1 Score for the CNN-LSTM was near 93%, whereas the baseline performance was around 58%. Unfortunately, the CNN-LSTM model was not able to optimize a curated list of products on each webpage as the dataset did not include an attribute that kept track of the items a customer searched on a webpage as Ilardi et. al. had done in their experiments.

# Future Works

RetailRocket dataset did not include an attribute that kept a record of the user’s searches and historical behavior on a webpage as Ilardi et. al. had initially proposed in their experiments; the proposed solution would need to be tested against a dataset where the model is provided a list of the user’s searches to determine whether the CNN-LSTM model does indeed outperform the LSTM model. Additionally, in terms of the dataset, we could incorporate categorical fields, whereas the RetailRocket dataset only used binary fields.

Transformer architectures have been shown to significantly outperform other neural networks when used for NLP cases; however, since transformers were recently introduced, further research is required to determine whether transformers can be used in recommendation systems. Generative Pre-Trained Transformer (GPT), Roberta, and Alberta – alternative transformer architectures – can also be used in place of the BERT transformer. ElMo architecture, a neural network based on bi-direction LSTM architecture, can also be tested to determine its compatibility with recommendation systems.

Overall, we did see a significant improvement in performance in terms of the F1 metric; however, the two-layered CNN-LSTM model took approximately 3,000 ms longer to train on a small dataset. Therefore, if the CNN-LSTM were to be trained on a larger dataset, the time required to train may not make the CNN-LSTM scalable to be used in production, especially with billions of users and products.

##### References

1. C. Lo, H. Yu, X. Yin, K. Shetty, C. He, K. Hu, J. M. Platz, A. Ilardi, and S. Madhvanath, “Page-level optimization of e-commerce item recommendations,” *Fifteenth ACM Conference on Recommender Systems*, 2021.
2. Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bender- sky, and Marc Najork. 2019. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. In Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval (Santa Clara, CA, USA) (ICTIR ’19). Association for Computing Machinery, New York, NY, USA, 85–92. https://doi.org/10.1145/3341981.3344218
3. X. Aubert. 2002. An overview of decoding techniques for large vocabulary continuous speech recognition. Comput. Speech Lang. 16 (2002), 89–114.
4. Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2019. Seq2Slate: Re-ranking and Slate Optimization with RNNs. arXiv:1810.02019 [cs.IR]
5. Walid Bendada, Guillaume Salha, and Théo Bontempelli. 2020. Carousel Personalization in Music Streaming Apps with Contextual Bandits. In Fourteenth ACM Conference on Recommender Systems (Virtual Event, Brazil) (RecSys ’20). Association for Computing Machinery, New York, NY, USA, 420–425. https: //doi.org/10.1145/3383313.3412217
6. Tianqi Chen and Carlos Guestrin. 2016. XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Aug 2016). <https://doi.org/10.1145/2939672.293978>
7. Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. 2009. Ranking Measures and Loss Functions in Learning to Rank. In Proceedings of the 22nd International Conference on Neural Information Processing Systems (Vancouver, British Columbia, Canada) (NIPS’09). Curran Associates Inc., Red Hook, NY, USA, 315–323.
8. Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (Boston, Massachusetts, USA) (RecSys ’16). Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/ 2959100.2959190
9. [9] Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang, Wenwu Ou, and Peng Jiang. 2019. A Pareto-Efficient Algorithm for Multiple Objective Optimization in e-Commerce Recommendation (RecSys ’19). Association for Computing Machinery, New York, NY, USA, 20–28. https: //doi.org/10.1145/3298689.3346998